UNIVERSITAT POLITÈCNICA DE CATALUNYA

DEPARTAMENT DE MATEMÀTICA APLICADA IV

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE DES SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

Doctor by the UPC

Programa de doctorat en
Matemàtica Aplicada

PhD of Science

of Université de Nice - Sophia Antipolis
Speciality: Computer Science

Defended by

Ignasi SAU VALLS

Optimization in Graphs under Degree Constraints Application to Telecommunication Networks

Grup de Grafs i Combinatòria
Advisor:
Xavier Muñoz López

Mascotte project, INRIA/CNRS-UNS
Advisors:
Jean-Claude Bermond, David Coudert

Defended on October 16, 2009

JURY:

Reviewers:	Pavol Hell	-	Simon Fraser Univ. (Vancouver, Canada)
	David Peleg	-	Weizmann Inst. of Science (Rehovot, Israel)
	Stéphan Thomassé	-	Univ. de Montpellier (Montpellier, France)
President :	Bruce B. Reed	-	McGill University (Montreal, Canada)
Examinators:	Jean-Claude Bermond	-	CNRS (Sophia-Antipolis, France)
	David Coudert	-	INRIA (Sophia-Antipolis, France)
	Xavier Muñoz	-	UPC (Barcelona, Catalonia, Spain)
	Miquel-Àngel Fiol	-	UPC (Barcelona, Catalonia, Spain)
Invited:	Stéphane Pérennes	-	CNRS (Sophia-Antipolis, France)

Acknowledgements

Thanks a lot to all my coauthors and scientific colleagues; this thesis is also theirs. A very special mention goes to my three advisors – Jean-Claude, David and Xavi – for their trust in me, their wise guidance and their constant generosity. I would like to mention also the members of Université de Nice-Sophia Antipolis, INRIA Sophia Antipolis, Universitat Politècnica de Catalunya and, more specifically, Departament de Matemàtica Aplicada IV.

Un grand merci à toute l'équipe MASCOTTE pour l'ambience et les conditions de travail (simplement exceptionelles), à mes camarades niçois, à l'ensemble de mes colocs (immense mais fini quand même) et aux superbes joueurs de futsal du CSPF pour leur accueil.

Muchas gracias a todos los responsables de que vaya a guardar para siempre un inmejorable recuerdo de mi etapa en Nice. Lo habéis intentado incansablemente día y noche durante los últimos cuatro años, y no hace falta que os diga que lo habéis conseguido. Yo creo que los *collectifs* y los packs de *kro* habrán tenido algo que ver... *plas plas plas*!

Muito obrigado também aos meus amigos brasileiros por os seus sinceros sorrisos.

Moltes gràcies als amics de (i/o per a) tota la vida: els del *col·le*, els de Ribes, els de la *uni*, ... prefereixo no posar-hi noms perquè segur que em deixaria algú i no m'ho perdonaríeu mai. Per molt lluny que estigui sempre us sento al meu costat. Hem compartit una infinitat de moments inesborrables, com els partits amb el *Todefiet* i l'*ETShiTB*, les Festes Majors amb els clàssics banys a la piscina, les nits que acaben la nit següent, els *duros* a l'Arcada, els partits de futbol ben *amanits*, "pozor!!", saltar des d'un avió, "bailarr", les memorables *naranjitades* i *pistoletades*, les entranyables nits a la sala d'estudis de la FME, la *Jumpy*, i tants d'altres que podríem anomenar junts.

I als meus avis i a tota la meva família. I al Barça per haver-me donat tantes alegries! Un record més que especial per a la Mar, que ha *patit* aquesta tesi tant o més que jo, i amb qui mica en mica – però amb pas ferm – anem descobrint països i sentiments.

Aquesta tesi va dedicada als meus pares, que estimo més que res en aquest món.

Barcelona
September 8, 2009

Optimization in Graphs under Degree Constraints. Application to Telecommunication Networks

Résumé : La première partie de cette thèse s'intéresse au groupage de trafic dans les réseaux de télécommunications. La notion de groupage de trafic correspond à l'agrégation de flux de faible débit dans des conduits de plus gros débit. Cependant, à chaque insertion ou extraction de trafic sur une longueur d'onde il faut placer dans le noeud du réseau un multiplexeur à insertion/extraction (ADM). De plus il faut un ADM pour chaque longueur d'onde utilisée dans le noeud, ce qui représente un coût d'équipements important. Les objectifs du groupage de trafic sont d'une part le partage efficace de la bande passante et d'autre part la réduction du coût des équipements de routage. Nous présentons des résultats d'inapproximabilité, des algorithmes d'approximation, un nouveau modèle qui permet au réseau de pouvoir router n'importe quel graphe de requêtes de degré borné, ainsi que des solutions optimales pour deux scénarios avec trafic all-to-all: l'anneau bidirectionnel et l'anneau unidirectionnel avec un facteur de groupage qui change de manière dynamique.

La deuxième partie de la thèse s'intéresse aux problèmes consistant à trouver des sous-graphes avec contraintes sur le degré. Cette classe de problèmes est plus générale que le groupage de trafic, qui est un cas particulier. Il s'agit de trouver des sous-graphes d'un graphe donné avec contraintes sur le degré, tout en optimisant un paramètre du graphe (très souvent, le nombre de sommets ou d'arêtes). Nous présentons des algorithmes d'approximation, des résultats d'inapproximabilité, des études sur la complexité paramétrique, des algorithmes exacts pour les graphes planaires, ainsi qu'une méthodologie générale qui permet de résoudre efficacement cette classe de problèmes (et de manière plus générale, la classe de problèmes tels qu'une solution peut être codé avec une partition d'un sous-ensemble des sommets) pour les graphes plongés dans une surface.

Finalement, plusieurs annexes présentent des résultats sur des problèmes connexes.

Mots-clés : Théorie des graphes, groupage de trafic, réseaux optiques, décomposition de graphes, optimisation, complexité, algorithmes d'approximation, complexité paramétrique, branchwidth, programmation dynamique, graphes dans les surfaces.

Optimization in Graphs under Degree Constraints. Application to Telecommunication Networks

Resum: La primera part d'aquesta tesi tracta del *traffic grooming*, un problema fonamental en xarxes òptiques. Consisteix en agrupar senyals de baixa velocitat en fluxos més ràpids, de cara a millorar la utilització de l'ample de banda i reduir el cost de la xarxa. L'objeciu és minimitzar el número d'*Add-Drop Multiplexers* (ADMs), uns aparells que insereixen/extrauen tràfic a/de fluxos d'alta velocitat. En termes de teoria de grafs, el problema es tradueix en trobar una partició d'un graf de *requests* en subgrafs amb un número fitat d'arestes, minimitzant el número total de vèrtexos de la partició. Al Capítol 1 s'estudia el cas on la topologia de la xarxa és un anell o un camí. Presentem el primer resultat de no-aproximabilitat del traffic grooming per valors fixats del *grooming factor* C, responent afirmativament una conjectura existent. També proposem un algorisme d'aproximació en temps polinomial amb un ràtio d'aproximació independent de C. Al Capítol 2 introduïm un nou model de traffic grooming en anells unidireccionals per dissenyar xarxes que puguin suportar *qualsevol* graf de requests amb grau màxim fitat. Fem palès que el problema és essencialement equivalent a trobar el més petit enter $M(C, \Delta)$ tal que les arestes de qualsevol graf amb grau màxim com a molt Δ es puguin particionar en subgrafs amb com a molt C arestes i que cada vèrtex aparegui en com a molt $M(C, \Delta)$ subgrafs, i trobem el valor de $M(C, \Delta)$ per gairebé tots els valors de C i Δ. El Capítol 3 tracta de traffic grooming en anells bidireccionals amb *routing* simètric i requests *all-to-all*. Presentem fites inferiors i famílies infinites de solucions òptimes per $C = 1, 2, 3$ i C de la forma $k(k+1)/2$. Al Capítol 4 estudiem el traffic grooming per xarxes òptiques de dos períodes, una variant del traffic grooming en anells unidireccionals amb dos grooming factors C i C' que permet un grau de dinamisme en el tràfic. Fent servir eines de particions de grafs, trobem el número mínim d'ADMs per $C = 4$ i $C' = 1, 2, 3$. L'estudi del traffic grooming condueix de manera natural a l'estudi d'una família de problemes en grafs amb restriccions genèriques sobre el grau dels vèrtexos. Aquest es el tema principal de la segona part de la tesi. Comencem estudiant al Capítol 5 la complexitat computacional d'aquests problemes, donant resultats de no-aproximabilitat i algorismes d'aproximació en temps polinomial. Al Capítol 6 analitzem la complexitat paramètrica de trobar subgrafs amb restriccions sobre el grau, quan el paràmetre és la mida del subgraf en qüestió. Demostrem resultats d'intractabilitat per grafs generals i proposem algorismes *fixed-parameter tractable* per grafs excloent un graf fixat com a menor. Al Capítol 7 proposem algorismes subexponencials en grafs planars per diverses famílies de problemes de subgrafs amb restriccions sobre el grau, fent servir teoria de *bidimensionalitat* i noves tècniques de programació dinàmica. Per acabar, al Capítol 8 presentem un *framework* per dissenyar algorismes basats en programació dinàmica per grafs en superfícies, amb una dependència *single exponential* en el *branchwidth*. El nostre mètode està basat en un nou tipus de *branch decomposition*, anomenada *surface cut decomposition*, que generalitza les *sphere cut decompositions* per grafs planars. L'existència d'aquests algorismes és deduïda fent servir diverses tècniques de teoria topològica de grafs i de combinatòria analítica.

Paraules clau: Teoria de grafs, traffic grooming, xarxes òptiques, particions de grafs, complexitat computacional, algorismes d'aproximació, complexitat paramètrica, branchwidth, programació dinàmica, grafs en superfícies.

Optimization in Graphs under Degree Constraints. Application to Telecommunication Networks

Abstract: The first part of this thesis is devoted to traffic grooming, which is a central problem in optical networks. It refers to packing low-rate signals into higher-speed streams, in order to improve bandwidth utilization and reduce the network cost. The objective is to minimize the number of Add-Drop Multiplexers (ADMs), which are devices that insert/extract low-rate traffic to/from a high-speed stream. In graph-theoretical terms, the problem can be translated into finding a partition of the edges of a request graph into subgraphs with bounded number of edges, the objective being to minimize the total number of vertices of the partition. We first focus in Chapter 1 on a general request graph when the topology is a ring or a path. We provide the first inapproximability result for traffic grooming for fixed values of the *grooming factor* C, answering affirmatively to a conjecture in the literature. We also provide a polynomial-time approximation algorithm for traffic grooming in rings and paths, with an approximation ratio independent of C. We introduce in Chapter 2 a new model of traffic grooming in unidirectional rings, in order to design networks being able to support *any* request graph with bounded maximum degree. We show that the problem is essentially equivalent to finding the least integer $M(C, \Delta)$ such that the edges of any graph with maximum degree at most Δ can be partitioned into subgraphs with at most C edges and each vertex appears in at most $M(C, \Delta)$ subgraphs, and we establish the value of $M(C, \Delta)$ for almost all values of C and Δ. In Chapter 3 we focus on traffic grooming in bidirectional rings with symmetric shortest path routing and all-to-all unitary requests, providing general lower bounds and infinite families of optimal solutions for $C = 1, 2, 3$ and C of the form $k(k+1)/2$. In Chapter 4 we study traffic grooming for two-period optical networks, a variation of the traffic grooming problem for WDM unidirectional ring networks with two grooming factors C and C' that allows some dynamism on the traffic. Using tools of graph decompositions, we determine the minimum number of ADMs for $C = 4$, and $C' = 1, 2, 3$. The study of the traffic grooming problem leads naturally to the study of a family of graph-theoretical problems dealing with general constraints on the degree. This is the topic of the second part of this thesis. We begin in Chapter 5 by studying the computational complexity of several families of degree-constrained problems, giving hardness results and polynomial-time approximation algorithms. We then study in Chapter 6 the parameterized complexity of finding degree-constrained subgraphs, when the parameter is the size of the subgraphs. We prove hardness results in general graphs and provide explicit fixed-parameter tractable algorithms for minor-free graphs. We obtain in Chapter 7 subexponential parameterized and exact algorithms for several families of degree-constrained subgraph problems on planar graphs, using bidimensionality theory combined with novel dynamic programming techniques. Finally, we provide in Chapter 8 a framework for the design of dynamic programming algorithms for surface-embedded graphs with single exponential dependence on branchwidth. Our approach is based on a new type of branch decomposition called *surface cut decomposition*, which generalizes sphere cut decompositions for planar graphs. The existence of such algorithms is proved using diverse techniques from topological graph theory and analytic combinatorics.

Keywords: Graph theory, traffic grooming, optical networks, graph partitioning, computational complexity, approximation algorithms, parameterized complexity, branchwidth, dynamic programming, graphs on surfaces.

Contents

Overview of this Thesis

This thesis contains the results obtained during the last three years in both MASCOTTE project of INRIA/CNRS-UNS in Sophia-Antipolis (France) and Departament de Matemàtica Aplicada 4 of Universitat Politècnica de Catalunya in Barcelona (Catalonia, Spain). It is also the result of numerous research visits, like the ones performed at Deparment of Theoretical Computer Science of IMFM (Ljubljana, Slovenia), the Computer Science Department of Technion (Haifa, Israel), Departamento de Computaçao of UFC (Fortaleza, Brazil), the Deparment of Mathematics of NKU (Athens, Greece), and Algorithms Research Group of University of Bergen (Bergen, Norway). These results would not have been obtained without the invaluable collaboration of my coauthors, to whom I am deeply indebted.

This thesis is organized as follows. We first provide in Part I some basic preliminaries and fix the notation to be used throughout the thesis. The concepts that are used only locally are defined in the corresponding chapters. There are two main parts (namely, Parts II and III, containing four chapters each, numbered from 1 to 8), each of them beginning with an introduction to the topic, an overview of the results in the literature, and a summary of our contributions. Part IV concludes the thesis and suggests some lines for further research. Finally, four appendices briefly summarize some further contributions that have not been included in the thesis.

The main topic of the thesis consists of graph optimization problems with constraints on the degree. Generally, these problems take as input a (weighted or unweighted) graph G and ask for a subgraph (or a set of subgraphs) of G satisfying certain degree constraints, while optimizing some parameter, usually the number of vertices or edges of the subgraphs. Most problems considered here are NP-hard, i.e., they are unlikely to be solvable by efficient exact algorithms. Therefore, it is important to design fast algorithms providing a solution which is provably close to the optimal solution. Nevertheless, sometimes it is possible to prove hardness results showing that certain algorithms cannot exist. Finally, it is also interesting to provide optimal solutions for some restricted classes of input graphs, and also to design *fast* (of course, not expected to run in polynomial time) exact or parameterized algorithms for a general input. The first part is devoted to traffic grooming.

Traffic grooming. Traffic grooming is a central problem in optical networks. Loosely speaking, it refers to packing low-rate signals into higher-speed streams, in order to improve bandwidth utilization and reduce network cost. The objective is to minimize the number of Add-Drop Multiplexers (ADMs), which are devices that insert/extract low-rate traffic

to/from a high-speed stream. In graph-theoretical terms, the problem can be translated into finding a partition of the edges of a request graph into subgraphs with bounded number of edges, the objective being to minimize the total number of vertices of the partition. The most used topology in real networks, like SONET WDM optical networks (see page 29), is the unidirectional or bidirectional ring.

We first focus in Chapter 1 on polynomial-time approximation algorithms and hardness results for a general request graph in the ring and path topologies. On the one hand, we provide the first inapproximability result for Ring and Path Traffic Grooming for fixed values of the grooming factor C (see page 29 for the definition), answering affirmatively to a conjecture in the literature. On the other hand, we provide a polynomial-time approximation algorithm for Ring and Path Traffic Grooming, based on a greedy cover algorithm, with an approximation ratio independent of C. This is the first approximation algorithm with this property, which is useful in practical applications since in backbone networks the grooming factor can be greater than the network size.

We introduce in Chapter 2 a new model of traffic grooming in unidirectional rings, in order to design networks being able to support *any* request graph with a fixed bounded degree. The existing theoretical models in the literature are much more rigid, and do not allow such adaptability. We show that the problem is essentially equivalent to finding the least integer $M(C, \Delta)$ such that the edges of any graph with maximum degree at most Δ can be partitioned into subgraphs with at most C edges and each vertex appears in at most $M(C, \Delta)$ subgraphs. We establish the value of $M(C, \Delta)$ for almost all values of C and Δ.

In Chapter 3 we focus on traffic grooming in bidirectional rings considering symmetric shortest path routing and all-to-all unitary requests, which had not been studied before. We formally state the problem, provide general lower bounds, and construct infinite families of optimal solutions for $C \in \{1, 2, 3\}$ and C of the form $k(k+1)/2$.

In Chapter 4 we study traffic grooming for two-period optical networks, a variation of the traffic grooming problem for WDM unidirectional ring networks that allows some dynamism on the traffic. In the two-period grooming problem, during the first period of time, there is an all-to-all uniform traffic among n nodes, each request using $1/C$ of the bandwidth; and during the second period, there is all-to-all uniform traffic only among a subset V of v nodes, each request now being allowed to use $1/C'$ of the bandwidth, where $C' < C$. Using tools of graph decompositions, we determine the minimum number of ADMs for any n, v and $C = 4$ and $C' \in \{1, 2, 3\}$.

As discussed above, the traffic grooming problem consists essentially in partitioning the edges of a graph into subgraphs with bounded number of edges, while minimizing the total number of vertices of the partition. In other words, the objective is to partition the edges of a graph into subgraphs maximizing the average *density* (defined as the edges-to-vertices ratio), or equivalently the average degree. The study of the traffic grooming problem leads naturally to the study of a family of graph-theoretic problems dealing with general constraints on the degree, such as the minimum degree or the maximum degree of the subgraphs. This is the topic of the second part of this thesis.

Degree-constrained subgraphs. For a typical degree-constrained subgraph problem, the objective is to find an optimal subgraph (usually, a subgraph with the maximum or

minimum number of vertices or edges) satisfying certain degree constraints, like bounded maximum or minimum degree.

We begin in Chapter 5 by studying the (classical) complexity of several families of degree-constrained problems, giving a variety of hardness results and polynomial-time approximation algorithms. Among others, we use the error amplification technique, probabilistic algorithms, and structural results of graph minors.

We then study in Chapter 6 the parameterized complexity of finding small degree-constrained subgraphs, when the parameter is the size of the subgraphs. We prove $W[1]$-hardness results in general graphs and provide explicit FPT algorithms (see page 24 for the definition of these terms) for H-minor-free graphs, using structural results and dynamic programming techniques.

Devising subexponential parameterized algorithms for degree-constrained subgraph problems on planar graphs is the topic of Chapter 7, which uses bidimensionality theory combined with novel dynamic programming techniques. As a result, we obtain subexponential parameterized and exact algorithms for several families of problems on planar graphs.

Finally, we provide in Chapter 8 a framework for the design of $2^{O(k)} \cdot n$ step dynamic programming algorithms for surface-embedded graphs on n vertices of branchwidth at most k. That way, we considerably extend the class of problems that can be solved by algorithms whose running times have a *single exponential dependence* on branchwidth, and improve the running time of several existing algorithms. Our approach is based on a new type of branch decomposition called *surface cut decomposition*, which generalizes sphere cut decompositions for planar graphs, and where dynamic programming should be applied for each particular problem. The existence of such algorithms is proved by a detailed analysis of how non-crossing partitions are arranged on surfaces with boundary and uses diverse techniques from topological graph theory and analytic combinatorics.

Further Contributions. The appendices of this thesis summarize several articles whose motivation mostly originated from the problems discussed so far. Each of these appendices contains just a succinct introduction to the topic and a brief description of the obtained results. The full proofs and all the details can be found in the corresponding articles.

Appendix A deals with permutation routing and (ℓ, k)-routing on plane grids. The packet routing problem plays an essential role in communication networks. It involves how to transfer data from some origins to some destinations within a reasonable amount of time. In the (ℓ, k)-routing problem, each node can send at most ℓ packets and receive at most k packets. In other words, the request graph can be represented by a bipartite graph where the two independent sets have degree bounded by ℓ and k, respectively. Permutation routing is the particular case $\ell = k = 1$. In Appendix A.1 we provide an optimal distributed permutation routing algorithm on full-duplex triangular grids, and in Appendix A.2 we provide tight permutation routing and (k, k)-routing algorithms on plane grids, as well as approximation algorithms for the general (ℓ, k)-routing problem.

Appendix B is concerned with the problem of routing a set of requests in AOLS networks with the aim of minimizing the number of labels required to ensure the forwarding.

This problem is in a sense similar to traffic grooming. In Appendix B.1 we study particularly this network design problem when the network is a path, providing an exact polynomial-time algorithm for the case in which all the requests have a common source and some approximation algorithms and heuristics for an arbitrary number of sources. In Appendix B.2 we formalize the considered problem by associating to each routing strategy a logical hypergraph whose hyperedges are dipaths of the physical graph, that correspond to tunnels in GMPLS terminology. Such a hypergraph is called a *hypergraph layout*, to which we assign a cost function given by its physical length plus the total number of hops traveled by the traffic. Minimizing the cost of the design of an AOLS network can then be expressed as finding a minimum cost hypergraph layout. We prove the first hardness results in this area and propose approximation algorithms for the problem, inspiring ourselves from techniques that had been previously applied to VPL problems for ATM networks.

Tolerance graphs are the topic of Appendix C. Tolerance graphs model interval relations in such a way that intervals can tolerate a certain degree of overlap without being in conflict. This subclass of perfect graphs has been extensively studied, due to both its interesting structure and its numerous applications. In Appendix C.1 we propose the first non-trivial intersection model for general tolerance graphs, given by three-dimensional parallelepipeds, which extends the widely known intersection model of parallelograms in the plane that characterizes the class of bounded tolerance graphs. This new representation also enables us to improve the time complexity of algorithms for computing a minimum coloring, a maximum clique, and a maximum weight independent set. The recognition of tolerance graphs – namely, the problem of deciding whether a given graph is a tolerance graph – as well as the recognition of their main subclass of bounded tolerance graphs, have been the most fundamental open problems on this class of graphs since their introduction in 1982. In Appendix C.2 we prove that both recognition problems are NP-complete, even in the case where the input graph is a trapezoid graph. The presented results are surprising because, on the one hand, most subclasses of perfect graphs admit polynomial recognition algorithms and, on the other hand, bounded tolerance graphs were believed to be efficiently recognizable as they are a natural special case of trapezoid graphs, which can be recognized in polynomial time.

The existence of a circuit through a prescribed set of vertices is an important graph-theoretical question. In Appendix D.1 we study the following problem: which is the largest integer k such that, given any subset of k ordered vertices of an infinite square grid, there exists a circuit visiting the k vertices in the prescribed order using each edge at most once? We prove that $k = 10$. To this end, we first provide a counterexample implying that $k < 11$. To show that $k \geq 10$, we introduce a methodology, based on the notion of *core graph*, to reduce drastically the number of possible vertex configurations, and then we test each one of the resulting configurations with an ILP solver.

A graph parameter is *self-dual* in some class of graphs embeddable in some surface if its value does not change in the dual graph more than a constant factor. Self-duality has been examined for several width-parameters, such as branchwidth, pathwidth, and treewidth (see page 21). In Appendix D.2 we give a direct proof of the self-duality of branchwidth in graphs embedded in some surface.

Finally, Appendix D.3 deals with a coloring problem in triangular grids. Namely, we are given an induced subgraph G of a triangular grid together with a integer demand function on its vertices. The objective is to assign to each vertex as many colors as its demand, in such a way that adjacent vertices get disjoint sets of colors, while minimizing the total number of used colors. This minimum is called the *multichromatic number* of G. Finding the multichromatic number of induced subgraphs of the triangular grid has important applications in cellular networks, and has been widely studied during the last years. We provide a simple algorithm to color any triangle-free induced subgraph of the triangular grid with at most seven colors when each vertex has demand three. Our result simplifies and improves some existing results in the literature.

The bibliography distinguishes between the personal publications of the author and other articles (tagged as "General Bibliography"). The personal bibliography is classified into "International Journals" (refs. [J1-J5]), "Book Chapters" (refs. [B6-B7]), "International Conferences" (refs. [C8-C21]), "National Conferences" (ref. [N22]), and "Submitted for Publication" (refs. [S23-S29]). Table 1 summarizes the coauthors associated to each chapter and appendix of this thesis, as well as the corresponding publications.

Chapter/Appendix	**Result of joint work with**	**Publications**
Chapter 1	O. Amini and S. Pérennes	[J1, C9, N22, B6]
Chapter 2	X. Muñoz and Z. Li	[C18, C15, S26]
Chapter 3	J.-C. Bermond and X. Muñoz	[C13, S25]
Chapter 4	J.-C. Bermond, C. J. Colbourn, L. Gionfriddo, and G. Quattrocchi	[J2]
Chapter 5	O. Amini, D. Peleg, S. Pérennes, and S. Saurabh	[C8, S23]
Chapter 6	O. Amini and S. Saurabh	[C10, S24]
Chapter 7	D. M. Thilikos	[C20, S28]
Chapter 8	J. Rué and D. M. Thilikos	[S27]
Appendix A.1	J. Žerovnik	[J5, C21]
Appendix A.2	O. Amini, F. Huc, and J. Žerovnik	[J3, B7]
Appendix B.1	J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, H. Rivano, and F. Solano	[C11]
Appendix B.2	J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, and F. Solano	[C12]
Appendix C.1	G. B. Mertzios and S. Zaks	[C16, J4]
Appendix C.2	G. B. Mertzios and S. Zaks	[C17]
Appendix D.1	D. Coudert and F. Giroire	[C14]
Appendix D.2	D. M. Thilikos	[C19]
Appendix D.3	P. Šparl and J. Žerovnik	[S29]

Table 1: Coauthors and publications associated to each part of this thesis.

Part I

Preliminaries

We provide here some basic preliminaries and fix the notation to be used throughout this thesis. This part is intended to be looked up when necessary, rather than to be read sequentially.

I.1 Graphs

We use standard graph-theoretical terminology, and we assume that the reader is familiar with the basic concepts of graph theory. For more details, see for instance the classical monograph of Berge [43] or the more recent book of Diestel [97].

Given a simple undirected graph $G = (V, E)$, and edge between the vertices u and v is denoted $\{u, v\}$, and then u and v are said to be *adjacent*. An edge is *incident* to its two endpoints. The *degree* of a vertex v in G is the number of vertices incident to v in G. Namely, $d(v) = |\{u \in V(G) : \{u, v\} \in E(G)\}|$. The *maximum degree* (resp. *minimum degree*) of a graph G is the maximum (resp. minimum) degree over all its vertices, and it is denoted $\Delta(G)$ (resp. $\delta(G)$).

For a graph $G = (V, E)$ and a subset $V' \subseteq V$, we denote the *induced subgraph* on V' by $G[V'] = (V', E')$, where $E' = \{\{u, v\} \in E : u, v \in V'\}$. For $v \in V$, we denote by $N_G(v)$ the *neighborhood* of v, namely $N_G(v) = \{u \in V : \{u, v\} \in E\}$. The *closed neighborhood* $N_G[v]$ of v is $N_G(v) \cup \{v\}$. In the same way we define $N_G[S]$ for $S \subseteq V$ as $N_G[S] = \cup_{v \in S} N_G[v]$, and $N(S) = N[S] \setminus S$. We may omit the subscript G if the graph is clear from the context.

A graph on n vertices is called *complete* if it contains an edge between each pair of vertices, and is denoted K_n. The complete graph on three vertices is known as the *triangle*. The *path* on n vertices $v_0, \ldots, v_{n-1}$ with the $n-1$ edges $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{n-2}, v_{n-1}\}$ is denoted P_n. The *cycle* on n vertices obtained from P_n by adding the edge $\{v_{n-1}, v_0\}$ is denoted C_n. A graph G is k-partite if $V(G)$ can be partitioned into k classes $V_0, \ldots, V_{k-1}$ such that there are only edges between classes V_i and V_j with $i \neq j$. The 2-partite (resp. 3-partite) graphs are known as *bipartite* (resp. *tripartite*). The *density* ρ of a graph $G = (V, E)$ is its edges-to-vertices ratio, that is $\rho(G) = \frac{|E(G)|}{|V(G)|}$. More generally, for any subset $S \subseteq V$, we denote its *density* by $\rho_G(S)$ or simply $\rho(S)$, and define it to be the density of the induced graph on S, i.e., $\rho(S) = \rho(G[S])$.

I.1.1 Tree-like decompositions of graphs

We define some well-known decompositions of graphs that, loosely speaking, quantify the resemblance of a graph to a tree.

Tree decompositions. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where $T = (I, F)$ is a tree, and $\mathcal{X} = \{X_i\},\ i \in I$ is a family of subsets of $V(G)$, called *bags* and indexed by the nodes of T, such that

1. each vertex $v \in V$ appears in at least one bag, i.e., $\bigcup_{i \in I} X_i = V$;
2. for each $v \in V$ the set of nodes indexed by $\{i \mid i \in I,\ v \in X_i\}$ forms a subtree of T;

3. For each edge $e = \{x, y\} \in E$, there is an $i \in I$ such that $x, y \in X_i$.

The *width* of a tree decomposition, denoted by $w((T, \mathcal{X}))$, is defined as $\max_{i \in I}\{|X_i| - 1\}$. The *treewidth* of G, denoted by $\mathbf{tw}(G)$, is the minimum width of a tree decomposition of G. We refer to the survey of Bodlaender [59] for an introductory overview on treewidth and its use in algorithmic graph theory.

Path decompositions. A tree decomposition $(T, \mathcal{X})$ in which T is a path is called a *path decomposition*. The width of a path decomposition $(T, \mathcal{X})$ is the width of $(T, \mathcal{X})$ as a tree decomposition. The *pathwidth* is defined exactly as above by restricting to the path decompositions, and it is denoted $\mathbf{pw}$. A path decomposition $(T, \mathcal{X})$, with T a path of length n, is usually denoted by $(X_0, X_1, \ldots, X_n)$.

Branch decompositions. Let G be a graph on n vertices. A *branch decomposition* (T, μ) of a graph G consists of an unrooted ternary tree T (i.e., all internal vertices are of degree three) and a bijection $\mu : L \to E(G)$ from the set L of leaves of T to the edge set of G. We define for every edge e of T the *middle set* $\mathbf{mid}(e) \subseteq V(G)$ as follows. Let T_1 and T_2 be the two connected components of $T \setminus \{e\}$. Then let G_i be the graph induced by the edge set $\{\mu(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$. The *middle set* is the intersection of the vertex sets of G_1 and G_2, i.e., $\mathbf{mid}(e) := V(G_1) \cap V(G_2)$. The *width* of (T, μ) is the maximum order of the middle sets over all edges of T, i.e., $\mathbf{w}(T, \mu) := \max\{|\mathbf{mid}(e)| : e \in T\}$. An optimal branch decomposition of G is defined by a tree T and a bijection μ which give the minimum width, the *branchwidth*, denoted by $\mathbf{bw}(G)$.

Robertson and Seymour [183] proved that the branchwidth and the treewidth of a graph G, with $|E(G)| \geq 3$, satisfy $\mathbf{bw}(G) \leq \mathbf{tw}(G) + 1 \leq \frac{3}{2}\mathbf{bw}(G)$. Therefore, a family of graphs has bounded branchwidth if and only if it has bounded treewidth.

I.1.2 Graph minors

Let $G = (V, E)$ be a simple undirected graph and let $e = \{x, y\} \in E$. We define $E_G(v) = \{\{v, u\} \mid u \in N_G(v)\}$. We denote by $G \backslash e$ the graph G' where $G' = (V, E - \{e\})$ and we say that G' *occurs from* G *after an edge removal*. We also denote by G/e the graph G' where

$$G' = (V - \{x, y\} \cup \{v_{x,y}\},\ E - E_G(x) - E_G(y) \cup \{\{v_{xy}, z\} \mid z \in N_G(x, y)\}),$$

where $v_{xy} \notin V$ is a new vertex, not in G. In this case we say that G' *occurs from* G *after an edge contraction*. If H occurs from G after a sequence of edge removals or contractions, we say that H is a *minor* of G, and that G is a *major* of H.

Given a graph H, a family of graphs $\mathcal{G}$ is *H-minor-free* is no graph in $\mathcal{G}$ contains H as a minor. For instance, planar graphs are K_5-minor-free and $K_{3,3}$-minor-free due to Kuratowski Theorem [43, 97].

Graph minors have been the topic of the so-called *graph minor theory*. This deep theory, mainly developed by Robertson and Seymour in a long series of papers, describes the structure of graphs with excluded minors. It culminates in the Graph Minors Theorem [188],

which states that every class of graphs closed under taking minors can be characterized by a finite set of excluded minors. Equivalently, it says that graphs are well-quasi ordered (WQO) by the minor relation, which means that in every infinite sequence or graphs there are two of them such that one is a minor of the other. The theory also has significant algorithmic consequences. Robertson and Seymour [186] proved that every class of graphs that is closed under taking minors (this means that if a graph G is inside the class, so is every minor of G) can be recognized in cubic time.

I.2 Computational Complexity

We provide in this section some basic definitions concerning basic classes, approximation algorithms and hardness of approximation to be freely used throughout this thesis. We assume that the reader is familiar with the classes P and NP, as well as with the asymptotic notation (like O, o, Ω, or Θ). For additional background material, the reader is referred to the books of Garey and Johnson [134] and Varizani [202].

I.2.1 Approximation algorithms

Given an NP-hard minimization (resp. maximization) problem Π and a polynomial time algorithm $\mathcal{A}$, let $OPT_\Pi(I)$ be the optimal value of the problem Π for the instance I, and let $ALG(I)$ be the value given by algorithm $\mathcal{A}$ for the instance I. We say that $\mathcal{A}$ is an *α-approximation algorithm* for Π if for any instance I of Π, $OPT_\Pi(I)/ALG(I) \geq \alpha$ (resp. $OPT_\Pi(I)/ALG(I) \leq \alpha$).

The class Apx consists of all NP-hard optimization problems that can be approximated within a constant factor. The subclass PTAS (Polynomial Time Approximation Scheme) contains the problems that can be approximated in polynomial time within a ratio $1+\varepsilon$ for *any* constant $\varepsilon > 0$. Assuming P $\neq$ NP, there is a strict inclusion of PTAS in Apx (for instance, Vertex Cover is in Apx \ PTAS), hence an Apx-hardness result for a problem implies the non-existence of a PTAS.

I.2.2 Hardness of approximation

For the inapproximability results presented in this thesis, we make use of the following reductions (c.f. for instance [202]).

For two minimization problems Π_1 and Π_2, a *gap-preserving reduction* from Π_1 to Π_2, parameterized by (f_1,α) and (f_2,β), where $\alpha,\beta : \mathbb{N}^+ \to \mathbb{N}^+$ and $f_1, f_2 : \{0,1\}^* \to \mathbb{R}^+$, is a procedure that given an instance x of Π_1, computes in polynomial time an instance y of Π_2 such that:

- if $OPT(x) \leq f_1(x)$, then $OPT(y) \leq f_2(x)$.
- if $OPT(x) > \alpha(|x|)f_1(x)$, then $OPT(y) > \beta(|x|)f_2(x)$.

The usefulness of gap-preserving reductions stems from the fact that if there is a gap-preserving reduction from Π_1 to Π_2 and it is NP-hard to approximate Π_1 within a factor strictly less than α, then it is also NP-hard to approximate Π_2 within a factor strictly less than β.

In general, inapproximability results are harder to proof than just NP-hardness. In particular, the existence of a PTAS is a difficult question to answer in general. For instance, in the case of the DENSE k-SUBGRAPH PROBLEM, whose best approximation ratio is $O(n^{\delta})$ for some $\delta < 1/3$ [114], the non-existence of a PTAS has been proved recently involving very technical details [159].

I.2.3 Parameterized complexity

Parameterized complexity is a recent approach to deal with intractable computational problems having some parameters that can be relatively small with respect to the input size. This area has been developed extensively during the last decade. The monograph of Downey and Fellows [103] provides a good introduction, and for recent developments see the books by Flum and Grohe [123] and by Niedermeier [174].

A *parameter* $\mathbf{P}$ is any function mapping graphs to non-negative integers. Examples of parameters are the size of a minimum vertex cover or the size of a maximum clique. The *parameterized problem* associated with parameter $\mathbf{P}$ asks, for some fixed k, whether $\mathbf{P}(G) \geq k$ for a given graph G.

For decision problems with input size n and parameter k, the goal is to design an algorithm with running time $f(k) \cdot n^{O(1)}$, where f depends only on k. Problems having such an algorithm are said to be *fixed-parameter tractable* (FPT). There is also a theory of parameterized intractability to identify parameterized problems that are unlikely to admit fixed-parameter tractable algorithms. There is a hierarchy of intractable parameterized problem classes above FPT, the most important ones being:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq M[2] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq XP$$

The principal analogue of the classical intractability class NP is $W[1]$, which is a strong analogue, because a fundamental problem complete for $W[1]$ is the k-STEP HALTING PROBLEM FOR NONDETERMINISTIC TURING MACHINES (with unlimited nondeterminism and alphabet size); this completeness result provides an analogue of Cook's Theorem in classical complexity. A convenient source of $W[1]$-hardness reductions is provided by the result stating that k-CLIQUE is complete for $W[1]$. The principal "working algorithmic" way of showing that a parameterized problem is unlikely to be fixed-parameter tractable, is to prove its $W[1]$-hardness using a parameterized reduction, which is defined as follows.

Let Π, Π' be two parameterized problems, with instances (x, k) and (x', k'), respectively. We say that Π is (uniformly many:1) *reducible* to Π' if there is a function Φ, called a *parameterized reduction*, which transforms (x, k) into $(x', g(k))$ in time $f(k) \cdot |x|^{\alpha}$, where $f, g : \mathbb{N} \to \mathbb{N}$ are arbitrary functions and α is a constant independent of k, so that $(x, k) \in \Pi$ if and only if $(x', g(k)) \in \Pi'$.

The notions of *kernel* and *kernelization* play a fundamental role in parameterized complexity. It captures data reduction taken from a fixed-parameter complexity point of view. Let $\mathcal{L}$ be a parameterized problem consisting of input (I,k), where I is the problem instance and k is the parameter. The term *reduction to a problem kernel* or *kernelization* means replacing instance (I,k) by a "reduced" instance (I',k') (called *problem kernel*) such that

1. $k' \leq k$ and $|I'| \leq g(k)$ for some function g only depending on k;
2. $(I,k) \in \mathcal{L}$ if and only if $(I',k') \in \mathcal{L}$; and
3. the reduction from (I,k) to (I',k') is computable in polynomial time in both $|I|$ and k.

The importance of kernels comes from the fact that a parameterized problem is fixed-parameter tractable if and only if it has a kernelization [103, 123, 174].

Minor closed parameters. We say that a parameter $\mathbf{P}$ is *minor closed* if whenever H is a minor of G, $\mathbf{P}(H) \leq \mathbf{P}(G)$. Examples of minor closed parameters are the size of a longest path or the size of a minimum feedback vertex set. A powerful algorithmic consequence of the Graph Minors Theorem [188] is that every minor closed parameterized problem is in FPT, i.e., it admits an algorithm running in $O(f(k) \cdot n^{O(1)})$ time. As we will discuss in more detail in Chapters 6-8, the drawback of this result is that the function $f(k)$ and the constants hidden in the big-Oh notation can be huge, and therefore these general FPT algorithms can be of limited practical value.

I.2.4 Some classical problems

For the sake of completeness, we provide here the definition of some classical problems (all NP-hard except Maximum Matching that is in P) that are mentioned in this thesis. For a complete list of classical NP-hard optimization problems, we refer the reader to [87, 134].

Maximum Matching
Input: A graph $G = (V,E)$.
Output: A subset $E' \subseteq E$ of the maximum size such that no two edges in E' share a common endpoint.

Maximum Clique
Input: A graph $G = (V,E)$.
Output: A subset $S \subseteq V$ of the maximum size such that there is an edge in E between any two vertices in S.

Maximum Independent Set
Input: A graph $G = (V,E)$.
Output: A subset $S \subseteq V$ of the maximum size such that there is no edge in E between any two vertices in S.

MINIMUM VERTEX COLORING
Input: A graph $G = (V, E)$.
Output: A function $f : V \to \{1, 2, \ldots, c\}$ such that $f(u) \neq f(v)$ for each edge $\{u, v\} \in E$ and such that c is minimized.

MINIMUM EDGE COLORING
Input: A graph $G = (V, E)$.
Output: A function $f : E \to \{1, 2, \ldots, c\}$ such that $f(e) \neq f(e')$ whenever e and e' share an endpoint and such that c is minimized.

MINIMUM VERTEX COVER
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of the minimum size such that for every edge $e = \{u, v\} \in E$, either $u \in S$ or $v \in S$.

MINIMUM FEEDBACK VERTEX SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of the minimum size such that $G[V \setminus S]$ has no cycles.

MINIMUM DOMINATING SET
Input: A graph $G = (V, E)$.
Output: A subset $S \subseteq V$ of the minimum size such that for every vertex $u \in V \setminus S$ there is a $v \in S$ such that $\{u, v\} \in E$.

LONGEST PATH
Input: A graph $G = (V, E)$.
Output: A path in G with the maximum number of edges.

LONGEST CYCLE
Input: A graph $G = (V, E)$.
Output: A cycle in G with the maximum number of edges.

DENSE k-SUBGRAPH (DkS)
Input: A graph $G = (V, E)$ and a positive integer k.
Output: A subset $S \subseteq V$, with $|S| = k$, such that $\rho(S)$ is maximized.

MINIMUM SET COVER
Input: A collection $\mathcal{C}$ of subsets of a finite set U.
Output: A subset $\mathcal{C}' \subseteq \mathcal{C}$ of the minimum size such that every element in U belongs to at least one member of $\mathcal{C}'$.

Part II

Traffic Grooming

II.1 Motivation

Optical *Wavelength Division Multiplexing* (WDM) is today the most promising technology to accommodate the explosive growth of Internet and telecommunication traffic in wide-area, metro-area, and local-area networks. Using WDM, the potential bandwidth of 50 THz of a fiber can be divided into multiple non-overlapping wavelengths or frequency channels. Since currently the commercially available optical fibers can support over a hundred frequency channels, such a channel has over one gigabit-per-second transmission speed. However, the network is usually required to support traffic connections at rates that are lower than the full wavelength capacity. In order to save equipment cost and improve network performance, it turns out to be very important to aggregate the multiple low-speed traffic connections, namely *requests*, into higher-speed streams. Traffic grooming is the generic term used to carry out this aggregation, in order to improve the usage of the bandwidth and of the components, and therefore to reduce the network cost.

When establishing a connection in an optical network, one has to install some equipments at both extremities of the connection, typically an optical transmitter (laser) at its source and an optical receiver at its destination. Due to the cost of building, installing, and maintaining devices, it is usually more interesting to use a single kind of device that may handle both transmission and reception, instead of two distinct devices. Such devices are called *Light Termination Equipment*, or LTE for short. Therefore, every connection is involved in two distinct LTEs, and two distinct connections may share the same LTE, provided that one ends at a node while the other originates at the same node. In this context, the traffic grooming problem refers to minimizing the number of LTEs that are needed in the network to serve all connection requests. The problem of minimizing the number of LTEs in the network being NP-hard [107,164], research effort has concentrated on the development of efficient approximation algorithms for both static and on-line traffic [88,110,121,122,192].

At another level of the network, traffic grooming refers in a more general sense to techniques used to combine low-speed traffic streams onto high-speed wavelengths in order to minimize the network-wide cost in terms of electronic switching. In this part of the thesis we focus on this version of traffic grooming. Nodes of the network insert and/or extract the data streams on a wavelength by means of *Add-Drop Multiplexers* (ADMs for short). A WDM optical network can handle many wavelengths, each with large bandwidth available (nowadays, in the order of 40 Gbps). On the other hand, a single user seldom usually does not need such large bandwidth. Therefore, by using multiplexed access such as *Time-Division Multiple Access* (TDMA) or *Code-Division Multiple Access* (CDMA), different users can share the same wavelength, thereby optimizing the bandwidth usage of the network. By using traffic grooming, not only the bandwidth usage is optimized, but also (and more importantly) the cost of the network can be cut by reducing the total number of ADMs. Such techniques become increasingly important for emerging network technologies, including SONET/WDM rings and MPLS/MPλS backbones [196], for which traffic grooming is essential.

In this context, one ADM is needed in a node each time we want to add or drop traffic from a wavelength at this node. Therefore, one has to place one ADM in a node for each wavelength in which traffic is added or dropped, as it is illustrated in Figure II.1. Here,

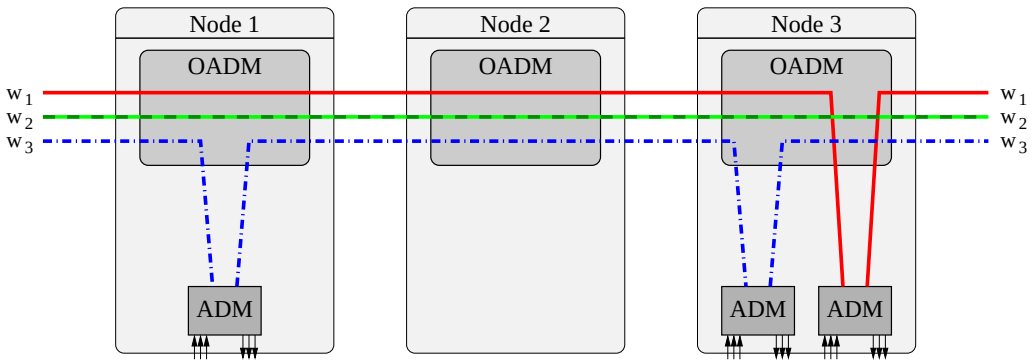

Figure II.1: Placement of ADMs in the network: one ADM for each wavelength used in a node.

the bandwidth requirement of a traffic stream is expressed as a fraction of the bandwidth offered by a single wavelength, which is called the *grooming factor* and henceforth denoted by C. Hence, an ADM is able to drop (resp. add) up to C unitary traffic streams from (resp. to) a given wavelength. Thus, the traffic grooming problem consists in minimizing the total number of ADMs to be installed in the network in order to accommodate all traffic streams.

The general traffic grooming problem with respect to the traffic requirement being NP-hard [71], recent works focus on specific issues. Most of the existing algorithms aim at grooming traffic in such a way that all the traffic between any given pair of nodes is carried on a minimum number of wavelengths. However, a large part of the network cost depends on the capacity of the multiplexing equipment required at each node. Hence, in order to minimize the overall network cost, algorithms have to take into account a trade-off between the number of wavelengths used and the number of required ADMs. Indeed, minimizing the number of ADMs may be incompatible with minimizing the number of wavelengths: the number of wavelengths and the number of ADMs cannot always be simultaneously minimized (see [51, 71, 135] for examples with unitary traffic). Both minimization problems have been considered by many authors. See for example the surveys [41, 105] for minimization of the number of wavelengths, [49, 135, 136, 151, 203, 208] for minimization of ADMs, and [150, 161] for on-line approaches. Numerical results, heuristics, and tables might be found in [51,204]. It makes also sense to aim at minimizing the number of *Optical Add-Drop Multiplexers* (OADMs for short, see Figure II.1), which are devices that are able to insert/extract entire wavelengths to/from an optical fiber [119]. The reader may also consult the following surveys [76, 106, 170, 209] and books [104, 194, 211] for other aspects of traffic grooming that are not considered here.

The sequel of this introduction to traffic grooming is structured as follows. We start in Section II.2 with a general definition of the traffic grooming problem and some examples. In Section II.3 we give an overview of the variants of traffic grooming that have been considered in this thesis, as well as our main contributions to each of these variants.

II.2 Problem Definition and Examples

We first give a precise description of LTE and ADM, and then we formalize the traffic grooming problem.

A Light Termination Equipment (LTE) is a device that realizes the interface between the optical and electronic domains. It is constituted of one optical receiver and one optical transmitter, so every connection involves two disctinct LTEs, one at each endpoint. We assume that the receiver and the transmitter of a LTE are tuned on the same wavelength (we would like to stress that other assumptions are technologically possible). Also, two distinct connections may share a LTE, provided that one ends at a node while the other originates at the same node, and that both connections are assigned the same wavelength.

An Add-Drop Multiplexer (ADM) is a device used in synchronous transmission networks (like SDH or SONET, see [104]) to add (insert) or drop (remove) lower-rate traffic channels from a higher-rate aggregated channel. In optical networks, each ADM contains a LTE to realize the interface between the optical domain (high-speed channel) and the electronic domain (lower-speed channels). Thus an ADM operates on a single high-speed data stream and therefore on a single wavelength, as can be seen in Figure II.1. The cost of an ADM is given by its capacity, that is, the maximum number of low-speed channels (provided that each of them have unitary bandwidth requirement) that can be added or dropped from the wavelength. The capacity of an ADM is called the grooming factor or grooming ratio C. Finally, remark that a LTE is a special case of an ADM for $C = 1$.

In optical networks with grooming capabilities, the traffic demands are expressed in terms of low-speed data channels. Thus, one has to assign to each connection request a path and a wavelength while respecting that at most C connection requests can be assigned the same wavelength on the same link of the network.

The precise statement of the traffic grooming problem depends on the particular assumptions, like the considered topology (for instance, a path or a ring) or the traffic pattern (for instance, an all-to-all setting or a bounded-degree request graph). This is the reason why we shall state the precise definition of each considered model in the corresponding chapter. For the sake of intuition, we provide here a general statement that does not capture the particularities of each setting.

A general instance of the Traffic Grooming problem is a triple (G, R, C), where $G = (V, E)$ is a digraph modeling the network topology, R is a set of connection requests and C is a positive integer, namely the grooming factor. Given a connection request $r \in R$ identified by a couple of nodes aiming to communicate, let P_r be the set of the directed paths in G connecting the two endpoints relative to r. There are two main issues to be addressed:

- the determination of a path system (or path assignment) of (G, R), that is a function $p : R \mapsto \bigcup_{r \in R} P_r$;
- the determination of a proper wavelength assignment of (G, R), that is a function $w : R \mapsto \mathbb{N}^+$ such that for any arc $e \in E$ at most C paths using e are assigned the same wavelength.

Every request $r \in R$ needs an ADM at each of its endpoint nodes. The key point is that the same ADM can be shared by the paths having a common endpoint which are assigned the same wavelength. The traffic grooming problem is the optimization problem of finding functions p, w for (G, R, C) minimizing the total number of used ADMs.

As we shall see in Chapters 1-4, the results presented in this thesis deal mainly with the second issue, that is, the assignment w of wavelengths to the requests. To fix ideas we provide now two examples, for unidirectional and bidirectional rings, respectively.

Unidirectional ring. Suppose we have a unidirectional ring with 4 nodes $\{1, 2, 3, 4\}$ and an all-to-all symmetric unitary traffic (i.e., one request between each couple of nodes). When the traffic requirement is symmetric, it can be easily shown (by exchanging wavelengths) that there always exists an optimal solution in which the same wavelength is given to each pair of symmetric requests. Thus without loss of generality we assign to each pair of symmetric requests, called a *circle*, the same wavelength. Then each circle uses $\frac{1}{C}$ of the bandwidth of the whole ring. If the two end-nodes of a circle are i and j, we need one ADM at node i and one at node j. The main point is that if two requests have a common end-node, they can share an ADM if they are assigned the same wavelength. In our example, there are therefore such 6 circles (i, j) for $1 \leq i < j \leq 4$. If there is no grooming (i.e., $C = 1$) we need 6 wavelengths (one per circle) and a total of 12 ADMs. If we have a grooming factor $C = 2$, we can put on the same wavelength two circles, using 3 or 4 ADMs according to whether they share an end-node or not. For example, we can put together $(1, 2)$ and $(2, 3)$ on one wavelength; $(1, 3)$ and $(3, 4)$ on a second wavelength, and $(1, 4)$ and $(2, 4)$ on a third one, for a total of 9 ADMs, and this is optimal. Now, if we allow a grooming factor $C = 3$, we can use only 2 wavelengths. Indeed, if we put together on one wavelength $(1, 2)$, $(2, 3)$, and $(3, 4)$ and on the other one $(1, 3)$, $(2, 4)$, and $(1, 4)$ we need 8 ADMs (first solution in Figure II.2); but we can do better by putting on the first wavelength $(1, 2)$, $(2, 3)$, and $(1, 3)$ and on the second one $(1, 4)$, $(2, 4)$, and $(3, 4)$, therefore using 7 ADMs (second solution in Figure II.2).

More formally, in the above example with 4 vertices and $C = 3$, the first solution consists in a partition of the edges of K_4 (each edge of K_4 corresponds to a circle) into two paths with four vertices each: $[1, 2, 3, 4]$ and $[1, 4, 2, 3]$, while the second solution corresponds to a decomposition into a triangle $(1, 2, 3)$ and a star with edges $(1, 4)$, $(2, 4)$, and $(3, 4)$ (see Figure II.2).

Bidirectional ring. Consider now a bidirectional ring on five nodes $\{0, 1, 2, 3, 4\}$ with all-to-all unitary traffic modeled by the complete symmetric digraph K_5^*. In this setting, it is better (in terms of the number of used ADMs) to route requests (i, j) and (j, i) on different wavelengths using shortest path routing. For example, with grooming factor $C = 3$, we can put on one wavelength the requests $\{(i, i + 1 \bmod 5), (i, i + 2 \bmod 5) : i = 0, \ldots, 4\}$ routed clockwise, and on another wavelength the requests $\{(i, i - 1 \bmod 5), (i, i - 2 \bmod 5) : i = 0, \ldots, 4\}$ routed counterclockwise. We need 5 ADMs on each wavelength, so overall 10 ADMs. But if requests (i, j) and (j, i) are routed on a same wavelength, then we can put at most 3 circles (pairs of symmetric requests) per wavelength, using at least 3 ADMs. Since K_5^* contains 10 circles, we need at least 4 wavelengths: 3 of them with 3 circles each

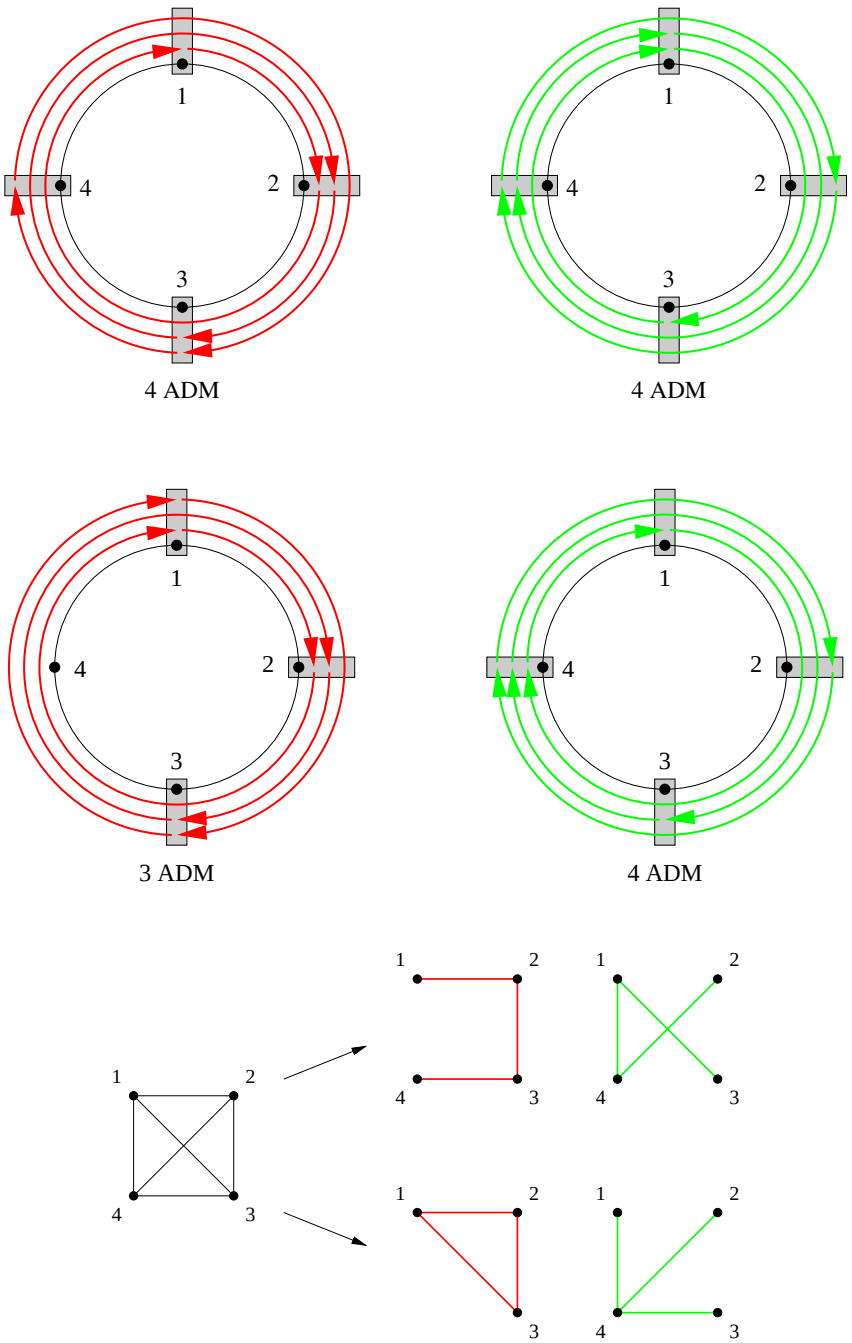

Figure II.2: Traffic grooming for a unidirectional ring with 4 nodes, grooming factor $C = 3$, and all-to-all unitary traffic. The above solution uses $4+4 = 8$ ADMs, whereas the second one uses $3+4 = 7$ ADMs. Below, the corresponding partitions of K_4 are illustrated.

(and therefore at least 3 ADMs each) and one of them with at least 1 circle and 2 ADMs, so overall at least 11 ADMs.

With grooming factor $C = 2$, we can put on one wavelength requests $\{(i, i+1 \bmod 5) : i = 0, \ldots, 4\}$ and on another wavelength requests $\{(i, i+2 \bmod 5) : i = 0, \ldots, 4\}$. Symmetric requests are routed similarly in the opposite direction. We obtain the first partition of Figure II.3, using overall $2 \cdot 10 = 20$ ADMs. But we can do better by putting on a first wavelength requests $\{(i, i+1 \bmod 5) : i = 0, \ldots, 4\}$, request $(0, 2)$, and request $(2, 4)$ using 5 ADMs, and on a second wavelength requests $(1, 3)$, $(3, 5)$, and $(4, 1)$ using 4 ADMs. We obtain the second partition of Figure II.3, using $2 \cdot (5 + 4) = 18$ ADMs overall.

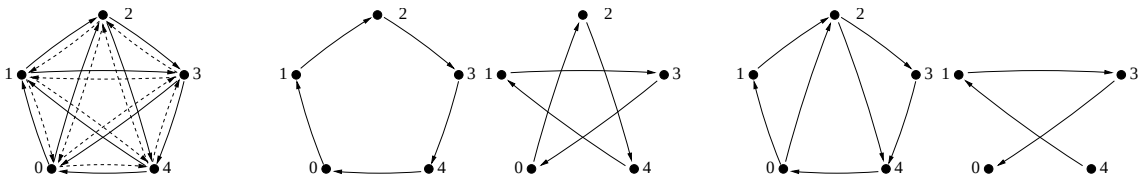

Figure II.3: On the left, a K_5^*. In the middle and on the right, two valid partitions of K_5^* when $C = 2$ using 10 and 9 ADMs, respectively. Symmetric requests are routed counterclockwise and partitioned similarly, hence using 20 and 18 ADMs, respectively.

II.3 State-of-the-art and our Contribution

The topic of traffic grooming is so large that it would be too ambitious to pretend to provide here a complete state-of-the-art. Most of the literature about traffic grooming originates from an engineering context (cf. for instance the books and surveys [B6, 76, 104, 106, 170, 194, 209, 211]), hence the vast majority of articles about traffic grooming propose heuristics that are then contrasted by the corresponding simulations. In this thesis we are only concerned with the theoretical aspects of traffic grooming, such as complexity results, approximation algorithms with a provable approximation ratio, or the construction of optimal solutions for restricted instances and topologies. It is far from the intention of the author to suggest that purely theoretical results are more important or more significant than applied ones; the fact that this thesis does not cover the literature originating from the engineering community is just a matter of scientific knowledge and interests.

We survey the existing results and contextualize our contributions concerning hardness and approximation, the all-to-all set of requests, and pseudo-dynamic scenarios in Sections II.3.1, II.3.2, and II.3.3, respectively.

II.3.1 Hardness and approximation

Most approximation algorithms and hardness results deal with the case where the topology is a ring or a path. This is because, on the one hand, these topologies are widely used in practical applications (like SONET WDM rings). On the other hand, whereas the traffic grooming problem can be clearly stated in paths and rings (see Chapter 1), the notion of

traffic grooming in topologies with vertices of degree at least three is somehow ambiguous, as it heavily depends on the technological devices used in each network. Approximation algorithms and NP-hardness results for a simplified model of traffic grooming in stars and trees can be found in [118]. Another model of traffic grooming in stars and trees is studied in [152]. From now on we focus on the case when the physical topology is a ring or a path.

Hardness results. The notion of traffic grooming with $C > 1$ was introduced in [136] for the ring topology. The problem has been proved to be NP-hard for ring networks and general C [71] using a reduction from the Bin Packing problem. Another proof was also mentioned in [203]. On the other hand, there was no result on the inapproximability of the problem for fixed $C \geq 1$. In [74] Chow and Lin conjectured that Ring Traffic Grooming is Max SNP-hard (or equivalently, Apx-hard, modulo PTAS-reductions) for any fixed value of the grooming factor. We answer affirmatively to this question in Theorem 1.3 of Chapter 1, providing the first hardness result for the Ring Traffic Grooming problem for fixed values of the grooming factor C.

Considering C as part of the input, in [152] it was proved that Path Traffic Grooming does not admit a constant-factor approximation unless P = NP. For fixed values of C, Path Traffic Grooming was proved to be in P for $C = 1$ [44], but the complexity for fixed $C \geq 2$ has been an open question for a while. Recently, it has been proved in [191] that Path Traffic Grooming for fixed $C > 1$ is NP-hard for *bounded number of wavelengths*. Our approach permits us to improve this result by proving the Apx-hardness of Path Traffic Grooming for any fixed $C > 1$ and *unbounded* number of wavelengths (see Theorem 1.5 in page 48). That is, we rule out the existence of a PTAS for fixed values of C, unless P=NP. In particular, this extends the NP-hardness result of [191] to the case where the number of wavelengths is not bounded.

The main ingredient of our approach is the proof of the Apx-completeness (given in Section 1.2 of Chapter 1) of the problem of finding the maximum number of edge-disjoint triangles in a tripartite graph with bounded degree B: Maximum B-Bounded Edge Covering by Triangles (MECT-B for short). The proof is obtained by L-reduction from Maximum Bounded Covering by 3-Sets, which was proved to be MAX SNP-complete in [155]. A simple modification of this technique permits us to prove the Apx-completeness of finding the maximum number of edge-disjoint odd cycles of given length in a graph. This later claim is then used to extend our results to arbitrary values of C, see Sections 1.2 and 1.3 of Chapter 1.

Approximation algorithms. Since Ring Traffic Grooming and Path Traffic Grooming are NP-hard, it is natural to devise polynomial-time approximation algorithms. We first focus on the ring topology.

As we discuss in Section 1.3 (page 44), it is trivial to obtain a $O(\sqrt{C})$-approximation with running time polynomial in both C and n (this fact was first proved in [138]), where n is the number of nodes of the network. For $C = 1$ (that is, for the minimization of LTEs, which is also known to be NP-hard [107,164]) the best algorithm in rings achieves an approximation ratio of 10/7 [110]. For this specific case of $C = 1$ we refer the reader to [44, 52, 110, 192].

For general C, the best approximation algorithm [120] achieves an approximation factor of $O(\log C)$ by using a classical Set Cover approach, but the problem is that the running time is exponential in C (that is, $n^{O(C)}$). Since in practical applications SONET WDM rings are widely used as backbone optical networks [106,170], the grooming factor is usually greater than the size of the network, i.e., $C \geq n$. For those networks, the running time of the algorithm of [120] becomes exponential in n. Thus, it turns out to be important to find good approximation algorithms with running time polynomial in both n and C. In Section 1.4 of Chapter 1 we provide such an approximation algorithm, considering C as part of the input. Our algorithm finds a solution of Ring Traffic Grooming that approximates the optimal value within a factor $O(n^{1/3}\log^2 n)$ for any $C \geq 1$. To the best of our knowledge, this is the first polynomial-time approximation algorithm for the Ring Traffic Grooming problem with an approximation ratio which does not depend on C. Although the performance of this algorithm seems not to be very good at first sight, in fact we conjecture that for the general instance of the problem it is not possible to get rid of a factor n^{δ}, for some constant $\delta > 0$ (see Conjecture 1.1 in page 53). We also show that the general scheme of the algorithm yields a $O(\log^2 n)$-approximation if the request graph excludes a fixed graph as minor, for example if it is planar or of bounded genus. The main theoretical contribution of this algorithm is to relate the Traffic Grooming problem to the widely studied Dense k-Subgraph problem [114].

Concerning the case of the path, both the algorithm of [120] (with approximation ratio $O(\log C)$ fox fixed values of C) and the algorithm we present in Chapter 1 (with approximation ratio $O(n^{1/3}\log^2 n)$ irrespective of C) also apply to the path topology with slight modifications.

II.3.2 The all-to-all case

An important special case is given by a unidirectional SONET ring with n nodes, grooming ratio C, and all-to-all uniform unitary traffic. This problem has been modeled as a graph partitioning problem in both [48] and [138]. In the all-to-all case the set of requests is modeled by the complete graph K_n. To a wavelength λ is associated a subgraph B_λ in which each edge corresponds to a pair of symmetric requests (that is, a circle in the terminology used in Section II.2) and each node to an ADM. The grooming constraint, i.e., the fact that a wavelength can carry at most C requests, corresponds to the fact that the number of edges $|E(B_\lambda)|$ of each subgraph B_λ is at most C. The cost corresponds to the total number of vertices used in the subgraphs, and the objective is therefore to minimize this number.

Traffic Grooming in the Unidirectional Ring with All-to-all Requests
Input: Two integers n and C.
Output: Partition $E(K_n)$ into subgraphs B_λ, $1 \leq \lambda \leq \Lambda$, s.t. $|E(B_\lambda)| \leq C$ for all λ.
Objective: Minimize $\sum_{\lambda=1}^{\Lambda} |V(B_\lambda)|$.

With this setting, optimal constructions have been obtained (using tools of graph and design theory [77]) for the cases $C = 3$ [45], $C = 4$ [48, 151], $C = 5$ [47], $C = 6$ [46],

$C = 7$ [78], and $C \geq n(n-1)/6$ [51]. Recently, good approximated solutions for any value of C have been presented in [50]. The all-to-all traffic case has been also studied on the path in [44]. See [49] for a survey of most of these results.

Nevertheless, the case when the physical network is a bidirectional ring has been much less studied in the literature from this graph partitioning perspective, mostly because the above simplified model cannot be applied anymore. A MILP formulation of the problem can be found in [151]. In [84] tools from design theory were applied to the bidirectional ring for the special case $C = 8$, without providing lower bounds to compare the proposed solutions to the optimal ones. Finally, a lower bound (independent of the routing of the requests) was given in [74].

In a bidirectional ring, requests are routed either clockwise or counterclockwise. We study in Chapter 3 the bidirectional ring with symmetric shortest path routing and all-to-all traffic. This is the first attempt to deal with traffic grooming in bidirectional rings using an approach similar to [49]. Using graph partitioning techniques and combinatorial designs, we formally state the problem, provide general lower bounds, and construct infinite families of optimal solutions for $C \in \{1, 2, 3\}$ and C of the form $k(k+1)/2$, as well as asymptotically optimal solutions. See Chapter 3 for the details.

II.3.3 Pseudo-dynamic scenarios

In this section we discuss two variants of the traffic grooming in order to allow more dynamic settings than the ones discussed so far. Chapters 2 and 4 are concerned with a changeable request graph and a changeable grooming factor, respectively. We observe that these scenarios are not completely dynamic, in the sense of considering any request graph and any grooming factor, but they incorporate a flexibility that did not exist in the theoretical models considered so far.

Variable grooming factor. Most of the papers on grooming deal with a single (static) traffic matrix. Some articles consider variable (dynamic) traffic, such as finding a solution which works for the maximum traffic demand [56,210], but all keep a fixed grooming factor. Recently, an interesting variation of the traffic grooming problem, called grooming for two-period optical networks, has been introduced by Colbourn, Quattrocchi, and Syrotiuk in [80, 81] in order to capture some dynamic nature of the traffic. Informally, in the two-period grooming problem each time period supports different traffic requirements. During the first period of time there is all-to-all uniform traffic among n nodes, each request using $1/C$ of the bandwidth; but during the second period there is all-to-all traffic only among a subset V of v nodes, each request now being allowed to use a larger fraction of the bandwidth, namely $1/C'$ where $C' < C$.

In [80, 81] the authors completely solved the cases when $C = 2$ and $C = 3$ ($C' = 1$ or 2). In Chapter 4 we determine the minimum drop cost for all $n \geq v \geq 0$, $C = 4$, and $C' \in \{1, 2, 3\}$. If we further restrict the solution to use the minimum number of wavelengths, it turns out that the optimal drop cost under this constraint may differ from the absolute optimum (see the last paragraph before Section 4.5 of Chapter 4). We also determine the optimal

drop cost that uses the minimum number of wavelengths for all $n \geq v \geq 0$, $C = 4$, and $C' \in \{1, 2, 3\}$. The precise cost formulas can be found in Section 4.1 of Chapter 4.

Variable request graph. As mentioned above, most of previous work on traffic grooming has focused on the case where the requests are given as input [J1, 48, 49, 106, 118, 120, 138, 170]. We consider in Chapter 2 the case where only the network topology is given, together with a bound Δ on the degree of the request graph. We would like to place, for each value of the grooming factor C, a minimum number of ADMs at each node in such a way that they could support *any* traffic pattern where each node is the end-node of at most Δ requests. This model is interesting because the network can support dynamic traffic without replacement of the ADMs. In other words, instead of placing the ADMs *a posteriori* for a given traffic demand, we would like to place them *a priori*.

From a practical point of view, it is interesting to design a network being able to support any request graph with maximum degree not exceeding a given constant. This situation is usual in real optical networks, since due to technology constraints the number of allowed communications for each node is usually bounded. This flexibility can also be thought from another point of view: if we have a limited number of available ADMs to place at the nodes of the network, then it is interesting to know which is the maximum degree of a request graph that our network is able to support, depending on the grooming factor. Equivalently, given a maximum degree and a number of available ADMs, it is useful to know which values of the grooming factor the network is able to support.

The aim of Chapter 2 is to provide a theoretical framework to design such networks with dynamically changing traffic. More precisely, we study the case when the physical network is given by a unidirectional ring. We show that the problem is essentially equivalent to finding the least integer $M(C, \Delta)$ such that the edges of any graph with maximum degree at most Δ can be partitioned into subgraphs with at most C edges and each vertex appears in at most $M(C, \Delta)$ subgraphs (see Section 2.1 for the details). We establish the value of $M(C, \Delta)$ for almost all values of C and Δ, leaving open only the case where $\Delta \geq 5$ is odd, $\Delta \pmod{2C}$ is between 3 and $C - 1$, $C \geq 4$, and the request graph does not contain a perfect matching (see Table 2.1 in page 71 for a summary of the results). For these open cases, we provide upper bounds that differ from the optimal value by at most one.

Chapter 1

Hardness and Approximation

In this chapter we focus on traffic grooming on ring and path topologies. On the one hand, we provide an inapproximability result for Traffic Grooming for fixed values of the grooming factor C, answering affirmatively to a conjecture of Chow and Lin [74]. More precisely, we prove that Ring Traffic Grooming for fixed $C \geq 1$ and Path Traffic Grooming for fixed $C \geq 2$ are Apx-complete, even if the maximum degree of the request graph is bounded by a small constant. That is, they do not admit a PTAS unless P = NP. Both results rely on the fact that finding the maximum number of edge-disjoint triangles in a tripartite graph (and more generally cycles of length $2C + 1$ in a $(2C + 1)$-partite graph of girth $2C + 1$) is Apx-complete.

On the other hand, we provide a polynomial-time approximation algorithm for Ring and Path Traffic Grooming, based on a greedy cover algorithm, with an approximation ratio independent of C. Namely, the approximation guarantee is $O(n^{1/3} \log^2 n)$ for any $C \geq 1$, n being the size of the network. This is useful in practical applications, since in backbone networks the grooming factor is usually greater than the network size. Finally, we improve this approximation ratio under some extra assumptions about the request graph.

Keywords: traffic grooming, optical networks, SONET ADM, approximation algorithms, Apx-hardness, PTAS.

1.1 Introduction

As already mentioned in Section II.1 (page 29), the most accepted criterion to reduce the equipment cost in WDM optical networks is to minimize the number of electronic terminations, which is unanimously considered as the dominant cost, rather than the number of wavelengths. SONET ring is the most widely used optical network infrastructure today. In these networks, a communication between a pair of nodes is done via a *lightpath*, and each lightpath uses an Add-Drop Multiplexer (ADM), i.e., an electronic termination, at

each of its two endpoints. If each request uses $1/C$ of the capacity of a wavelength, C is said to be the *grooming factor*. We recall that the problem is equivalent to assigning a wavelength to each request in such a way that for any wavelength and any link of the network, there can be at most C requests using this link on this wavelength. The aim is to minimize the total number of ADMs.

Statement of the problem. In the graph-theoretical approach that we use, the set of requests is modeled by a graph R, and each vertex in the subgraph of R corresponding to a wavelength represents an ADM. The problem, in the case where the communication network is a ring, can be formally stated as follows.

Ring Traffic Grooming
Input: A cycle C_n on n vertices (network), a graph R (set of requests) on vertices of C_n, and a grooming factor C.
Output: Find for each edge $r = \{x, y\}$ of R, a path $P(r)$ in C_n between x and y, and a partition of the edges of R into subgraphs R_λ, $1 \leq \lambda \leq \Lambda$, such that for each edge e in $E(C_n)$ and for all λ, the number of paths $P(r)$ using e, r being an edge of R_λ, is at most C.
Objective: Minimize $\sum_{\lambda=1}^{\Lambda} \lvert V(R_\lambda)\rvert$.

The number of paths $P(r)$ using an edge $e \in E(C_n)$ in a given subgraph R_λ is known as the *load* of e in R_λ. That is, the load of the edges in any subgraph of the partition of $E(R)$ can be at most C. The statement of Path Traffic Grooming is analogous, replacing *cycle* C_n with *path* P_n. To fix ideas, consider a ring on five nodes and the complete graph of Figure 1.1 as request graph, and let $C = 2$. We exhibit two valid solutions of the problem, both using two subgraphs (i.e., two wavelengths). The lower solution is better because it uses 9 vertices instead of 10.

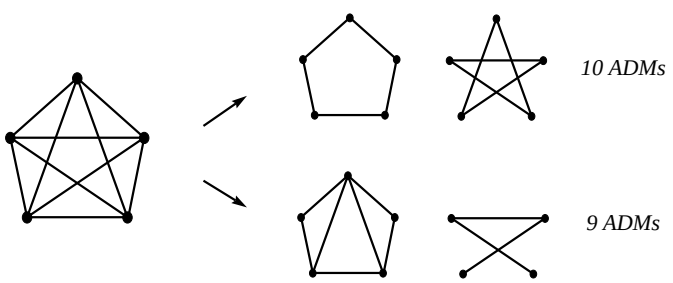

Figure 1.1: Two valid partitions of K_5 when $C = 2$, using different number of ADMs.

Our contribution. As discussed in Section II.3.1 (page 34), there was no result in the literature on the inapproximability of Ring Traffic Grooming for fixed $C \geq 1$. In [74] Chow and Lin conjectured that Traffic Grooming is Max SNP-hard (or equivalently, Apx-hard, modulo PTAS-reductions) for any fixed value of the grooming factor. We

answer affirmatively to this question in Theorem 1.3, providing the first hardness result for the Ring Traffic Grooming problem for fixed values of the grooming factor C.

Considering C as part of the input, in [152] it was proved that Path Traffic Grooming does not admit a constant-factor approximation unless P = NP. For fixed values of C, Path Traffic Grooming was proved to be in P for $C = 1$ [44], but the complexity for fixed $C \geq 2$ has been an open question for a while. Recently, it has been proved in [191] that Path Traffic Grooming for fixed $C > 1$ is NP-hard for *bounded number of wavelengths*. Our method permits us to improve this result in Section 1.3, by proving the Apx-hardness of Path Traffic Grooming for any fixed $C > 1$ and *unbounded* number of wavelengths. In particular, this extends the NP-hardness result of [191] to the case where the number of wavelengths is not bounded.

The main ingredient of our approach is the proof of the Apx-completeness (given in Section 1.2) of the problem of finding the maximum number of edge-disjoint triangles in a tripartite graph with bounded degree B: Maximum B-Bounded Edge Covering by Triangles (MECT-B for short). The proof is obtained by L-reduction from Maximum Bounded Covering by 3-Sets, which was proved to be MAX SNP-complete in [155]. A simple modification of this technique permits us to prove the Apx-completeness of finding the maximum number of edge-disjoint odd cycles of given length in a graph. This later claim is then used to extend our results to arbitrary values of C, see Sections 1.2 and 1.3.

The design of approximation algorithms for Traffic Grooming is the topic of the second part of this chapter. We present the results for the ring topology, but the same algorithm works also for the path topology. As we show in Section 1.3, it is trivial to obtain a $O(\sqrt{C})$-approximation with running time polynomial in C and n. For $C = 1$, the best algorithm in rings achieves an approximation ratio of 10/7 [110]. For general C, the best approximation algorithm [120] achieves an approximation factor of $O(\log C)$, but the problem is that the running time is exponential in C (that is, $n^{O(C)}$). Since in practical applications SONET WDM rings are widely used as backbone optical networks [106, 170], the grooming factor is usually greater than the size of the network, i.e., $C \geq n$. For those networks, the running time of this algorithm becomes exponential in n. Thus, it turns out to be important to find good approximation algorithms with running time polynomial in both n and C. In Section 1.4 we provide such an approximation algorithm, considering C as part of the input. Our algorithm finds a solution of Ring Traffic Grooming that approximates the optimal value within a factor $O(n^{1/3} \log^2 n)$ for any $C \geq 1$. To the best of our knowledge, this is the first polynomial-time approximation algorithm for the Ring Traffic Grooming problem with an approximation ratio which does not depend on C. Although the performance of this algorithm seems not to be very good at first sight, in fact we conjecture that for the general instance of the problem it is not possible to get rid of a factor n^{δ}, for some constant $\delta > 0$. Finally, we show that the general scheme of the algorithm yields a $O(\log^2 n)$-approximation if the request graph excludes a fixed graph as a minor, for example if R is planar or of bounded genus. The main theoretical contribution of the second part of this chapter is to relate the Traffic Grooming problem to the Dense k-Subgraph problem [114]. We conclude by proposing some further research directions to better understand the complexity of Traffic Grooming.

1.2 Apx-completeness of MECT-B

The problem of finding the maximum number of node- or edge-disjoint cycles in an undirected graph G has several applications, for instance in computational biology [40]. It is often the case that both the maximum degree of G and the length of the cycles to be found are bounded by a constant. In this section we are interested in the following problem:

MAXIMUM B-BOUNDED EDGE COVERING BY TRIANGLES (MECT-B)
Input: An undirected graph G with maximum degree at most B.
Objective: Find the maximum number of edge-disjoint triangles in G.

MECT-B is long known to be NP-hard [149], and the APX-hardness when requiring node-disjoint triangles was proved in [155]. Following the ideas of [155], in [64] it was proved that MECT-5 is APX-hard for general graphs and NP-hard for planar graphs. Finally, in [167] MECT-B was studied from a parameterized view, considering the number of edge-disjoint triangles as the parameter. Namely, is was proved that MECT-B is FPT by achieving a linear kernel (see [103]).

In this chapter we prove that MECT-B remains APX-hard for tripartite graphs. For convenience, we prove the MAX SNP-hardness of MECT-B, which is known to be the same as the APX-hardness modulo PTAS-reductions [202]. MECT-B is trivially in APX, since a simple greedy algorithm provides a 3-approximation. The best approximation guarantee for MECT-B is a $(3/2+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$ [153]. We need to introduce two problems to be used in the proof of Theorem 1.1: MAXIMUM BOUNDED COVERING BY 3-SETS (MAX 3SC-B for short): Given a collection of 3-subsets of a given set, each element appearing in at most B subsets, find the maximum number of disjoint subsets; and MAXIMUM BOUNDED INDEPENDENT SET (INDEP. SET-B for short): Given a graph of maximum degree $\leq B$, find a maximum independent set.

Theorem 1.1 MECT-B, $B \geq 10$, *is* APX*-complete for tripartite graphs.*

Proof: L-reduction from MAX 3SC-B and L-reduction to INDEP. SET-B.
We define h : MECT-B $\rightarrow$ INDEP. SET - (3/2(B-2)) as follows: given a graph G as instance I of MECT-B, we define the following instance $h(I)$ of INDEP. SET - (3/2(B-2)): the graph $h(G)$ contains a node v_T for every triangle T in G. There is an edge $\{v_{T_0}, v_{T_1}\}$ in $h(G)$ if and only if T_0 and T_1 share an edge in G. Given a solution A of $h(I)$, we define a solution $S_h(A)$ of I by taking the triangles corresponding to nodes in A. It is easily verified that (h, S_h) is an L-reduction.

We define f : MAX 3SC-B $\rightarrow$ MECT-(3B+1) in the following way: suppose that we are given as instance I, a collection $\mathcal{S}$ of 3-element subsets of a set X such that every element of X belongs to at most B members of $\mathcal{S}$. The problem for I consists in finding the maximal number $OPT(I)$ of disjoint subsets in $\mathcal{S}$. We construct an instance $f(I)$ of MECT-(3B+1), i.e., we construct a graph $G = (V, E)$ in which we ask for the maximum number $OPT(f(I))$ of edge-disjoint triangles. Let $\mathcal{S} = \{c_1, \ldots, c_r\}$, with $|c_i| = 3$. The local replacement f substitutes for each element $c_i = \{x, y, z\} \in \mathcal{S}$, the graph $G_i = (V_i, E_i)$ depicted in Figure 1.2.

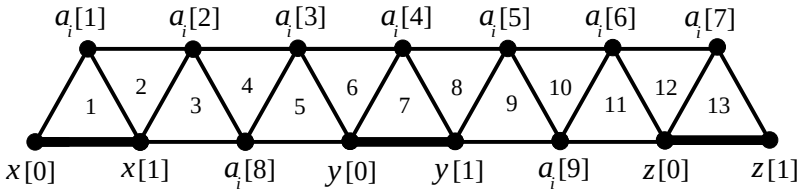

Figure 1.2: Gadget G_i used in the reduction of the proof of Theorem 1.1.

To avoid confusion, note by t any element in c_i, i.e., $t \in \{x, y, z\}$. Note that, for each element t, the nodes $t[0]$ and $t[1]$, and the edge $\{t[0], t[1]\}$ (corresponding to the thick edges in Figure 1.2) appear only once in G, regardless of the number of occurrences of t. On the other hand, we add 9 new vertices $a_i[j]$, $1 \leq j \leq 9$ for each subset c_i, $1 \leq i \leq |\mathcal{S}|$. More precisely, $G = (V, E) = \cup_{i=1}^{|\mathcal{S}|} G_i$, where $V = \bigcup_{t \in X} \{t[i] : i = 0, 1\} \cup \bigcup_{i=1}^{|\mathcal{S}|} \{a_i[j] : 1 \leq j \leq 9\}$ and $E = \bigcup_{i=1}^{|\mathcal{S}|} E_i$.

Given a solution A of $f(I)$ of size s_2, we modify it in polynomial time to another equal or better solution A' in the following way: in each G_i, if the three triangles covering the edges $\{x[0], x[1]\}$, $\{y[0], y[1]\}$, and $\{z[0], z[1]\}$ (numbered $1, 7, 13$ in Figure 1.2) belong to A, we choose the seven *odd* triangles of G_i to belong to A'. If not, we take the six *even* triangles. Let $s_2' \geq s_2$ be the size of A'. Then, we define a solution $S_f(A)$ of I by choosing the subset c_i to be in $S_f(A)$ if and only if A' contains exactly 7 triangles in G_i. We claim that the pair (f, S_f) is an L-reduction: in each G_i there are 13 different triangles, numbered from 1 to 13 in Figure 1.2. The only way to choose 7 edge-disjoint triangles in G_i is by taking all the *odd* triangles, and thus by covering the three edges $\{x[0], x[1]\}$, $\{y[0], y[1]\}$, and $\{z[0], z[1]\}$. All other choices of triangles yield at most 6 edge-disjoint triangles. The key observation is that we are able to choose 7 triangles exactly $OPT(I)$ times. Indeed, each time we choose 7 triangles we cover the edges corresponding to 3 elements of c_i, and since the number of disjoint c_i's in $\mathcal{S}$ is $OPT(I)$, this can be done exactly $OPT(I)$ times. On the other hand, one can easily see that $OPT(I) \geq \frac{|\mathcal{S}|}{3B}$. Hence:

$$\begin{aligned} OPT(f(I)) &= 7 \cdot OPT(I) + 6(|\mathcal{S}| - OPT(I)) \leq OPT(I) + 18B \cdot OPT(I) \\ &= (18B + 1)OPT(I). \end{aligned}$$

To conclude, note that if the solution $S_f(A)$ of I has size s_1, we have $OPT(I) - s_1 \leq OPT(f(I)) - s_2$. To see this, we observe that $OPT(f(I)) = 6r + OPT(I)$, and also $s_2' = 6r + s_1$, and so $OPT(f(I)) - OPT(I) = s_1 - s_2' \leq s_1 - s_2$.

Both (f, S_f) and (h, S_h) are L-reductions and MAX 3SC-B, $B \geq 3$ and Indep. Set-B, $B \geq 5$ are Max SNP-complete [155]. Thus, MECT-B, $B \geq 10$ is Max SNP-complete. Finally, note that the graph $G = (V, E)$ used in the proof is tripartite, where the vertex sets V_0, V_1, V_2 defining the tripartition are:

$$V_0 = \bigcup_{t \in X}^{|X|} t[0] \cup \bigcup_{i=1}^{|\mathcal{S}|} \{a_i[2], a_i[5]\}, \quad V_1 = \bigcup_{i=1}^{|\mathcal{S}|} \{a_i[j] : j = 1, 4, 7, 8, 9\},$$

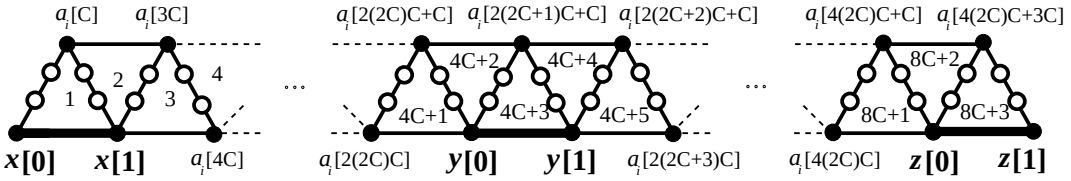

Figure 1.3: Adding $C-1$ inner points (depicted as $\circ$ in the figure) to prove the Apx-completeness of finding edge-disjoint C_{2C+1}'s.

$$V_2 = \bigcup_{i=1}^{|X|} t[1] \ \cup \ \bigcup_{t \in X}^{|\mathcal{S}|} \{a_i[3], a_i[6]\}.$$

□

The proof of the Apx-hardness of MECT-B of Theorem 1.1 can be extended to obtain the Apx-completeness of the problem of finding the maximum number of edge-disjoint cycles of length $2C+1$ for any fixed $C \geq 1$, as stated in the following theorem.

Theorem 1.2 *Let $\mathcal{G}$ be the class of $(2C+1)$-partite graphs G of girth $2C+1$, consisting of $(2C+1)$ parts $A_0, \ldots, A_{2C}$ such that the only edges are between A_i and $A_{i+1} \pmod{2C+1}$, $i = 0, \ldots, 2C$, and such that all the graphs induced by $V(G) \setminus A_i$ in G, for all $i = 0, \ldots, 2C$, form a forest. Then the problem of finding the maximum number of edge disjoint C_{2C+1}'s is* Apx*-complete in $\mathcal{G}$.*

Proof: First, note that a greedy algorithm provides a constant factor approximation with factor $2C+1$, so the problem is in Apx. Consider the gadget of the proof of Theorem 1.1 (see Figure 1.2). We modify this gadget in such a way that the same proof holds for C_{2C+1}'s instead of C_3's (triangles), and such that all the conditions of the theorem are verified. Given $C > 1$, we add a chain of $4C+1$ triangles between any two pair of triangles corresponding to *thick* edges (that is, between the edges corresponding to elements of X). Then we add $C-1$ inner points to all the edges going from up to down in the triangles. An example if shown in Figure 1.3.

It is easily seen that the graph built in this way is $(2C+1)$-partite. Indeed, it admits a partition into $(2C+1)$ parts, which consist of enumerating the vertices cyclically. Let $A_0, \ldots, A_{2C}$ be the different parts. In such a $(2C+1)$-partition, for any element $t \in X$, the vertex $t[0]$ belongs to A_0, and the vertex $t[1]$ belongs to A_{2C}. We need this property to ensure the consistency of our gadget when an element appears in more that one subset. Note that the graphs induced by $V(G) \setminus A_i$ in G, for all $i = 0, \ldots, 2C$, form a forest. At this point, one can rewrite the proof of Theorem 1.1 to obtain the result, just by changing the multiplicative constants. □

1.3 Apx-completeness of Traffic Grooming

In this section we prove the hardness results for Ring Traffic Grooming and Path Traffic Grooming. First we prove that Ring Traffic Grooming belongs to Apx

when C is fixed (i.e., not part of the input). The same result holds for Path Traffic Grooming.

Lemma 1.1 Ring Traffic Grooming *belongs to* Apx *for any fixed* $C \geq 1$.

Proof: To see that Ring Traffic Grooming is in Apx for any fixed $C \geq 1$, we have to find a constant-factor approximation algorithm. We use the fact that the best possible density ρ^* of any subgraph involved in the partition of the request graph in the ring is $O(\sqrt{C})$, given by a complete graph inducing load C in the edges of the ring (it is clear that no graph has greater density than the complete graph). We prove that the cost A of any solution $R_1, \ldots, R_\Lambda$ is in the interval $[\frac{|E(R)|}{\rho^*}, 2|E(R)|]$. This clearly implies that any solution has cost at most $2\rho^* = O(\sqrt{C})$ times the optimal cost. To see this, note that each edge of R contributes at most twice to the cost, so $A \leq 2|E(R)|$. On the other hand, we have

$$A = \sum_{\lambda=1}^{\Lambda} |V(R_\lambda)| = \sum_{\lambda=1}^{\Lambda} \frac{|E(R_\lambda)|}{\rho(R_\lambda)} \geq \sum_{\lambda=1}^{\Lambda} \frac{|E(R_\lambda)|}{\rho^*} = \frac{|E(R)|}{\rho^*}.$$

Thus, a $O(\sqrt{C})$-approximation is obtained just by taking *any* partition of the request graph. □

Since we will deal with tripartite graphs in the proof of Theorem 1.3, we need first a technical lemma concerning the structure of the optimal solutions of Ring Traffic Grooming in tripartite request graphs.

Lemma 1.2 *Let R be a tripartite instance graph of* Ring Traffic Grooming *for $C = 1$ such that the vertices belonging to the same class of the tripartition are placed consecutively in the ring, and let t^* be the maximum number of edge-disjoint triangles in R. If there exists a partition of $E(R)$ into triangles and P_4's which uses exactly t^* triangles, then this partition is optimal. The same property holds for* Path Traffic Grooming *and $C = 2$.*

Proof: We focus first on Ring Traffic Grooming. Let t^* the maximum number of edge-disjoint triangles of a partition of $E(R)$. When R is tripartite and $C = 1$, it is clear that the only possible subgraphs that can be involved in a partition of $E(R)$ are K_3, P_2, P_3, and P_4 (see Figure 1.4(a)). Since these three paths have density at most 3/4 (attained by the P_4), the cost A_t of any solution using t triangles satisfies

$$A_t \;\geq\; t + 4 \cdot \frac{|E(R)| - 3t}{3} = \frac{4}{3}|E(R)| - 3t \;\geq\; \frac{4}{3}|E(R)| - 3t^*. \tag{1.1}$$

Note that the above bound does not depend on t, and therefore holds for any solution. A partition as stated in the conditions of the lemma attains this lower bound, hence it is optimal. The same argument applies to the path and $C = 2$ (see Figure 1.4(b)), since the same subgraphs are involved in any partition. □

We are ready to state the main result of this section.

Theorem 1.3 Ring Traffic Grooming *is* Apx*-complete for fixed $C = 1$, even if the request graph has degree bounded by a constant $B \geq 10$. Thus, it does not admit a* PTAS *unless* P = NP.

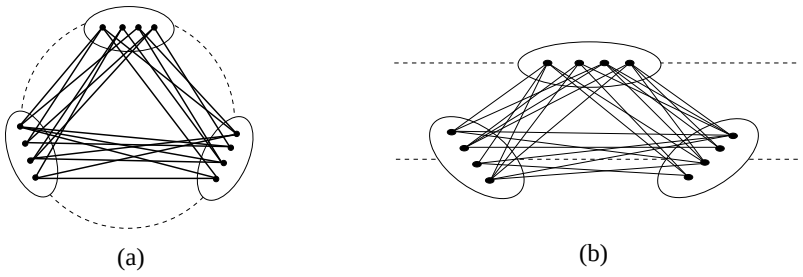

Figure 1.4: Tripartite request graphs used in Lemma 1.2: (a) in the ring for $c = 1$; (b) in the path for $C = 2$.

Proof: The problem is in Apx by Lemma 1.1. To prove the Apx-hardness, we consider the family of request graphs $\mathcal{R}$ defined as follows.

Mimic the proof of Theorem 1.1 replacing the gadget of Figure 1.2 with the gadget of Figure 1.5(a). With slight abuse of notation, the edge corresponding to an element x is also denoted x. It is easy to check that the same proof carries over to these new gadgets, and therefore the problem of finding the maximum number of edge-disjoint triangles in this class $\mathcal{R}$ of graphs is Apx-hard. Note that all the graphs built in this way are also tripartite, as shown in Figure 1.5(a).

Håstad proved [145] that Maximum Bounded Covering by 3-Sets is Apx-hard even restricted to instances for which we know that there exists a collection of mutually disjoint 3-subsets covering *all* the elements in the set. Therefore, we can assume without loss of generality that any optimal solution of MECT-B in a graph $R \in \mathcal{R}$ corresponds to a collection of mutually disjoint 3-subsets covering *all* the elements in the set. Hence, such an optimal solution of MECT-B restricted to each gadget G_i corresponding to the set $c_i = \{x, y, z\}$ satisfies:

(i) either it contains the three edges x, y, z corresponding to the elements in the set $c_i = \{x, y, z\}$; or

(ii) it contains *none* of the edges x, y, z.

Thinking of the graphs $R \in \mathcal{R}$ as instances of Ring Traffic Grooming, the key observation is that:

- in case *(i)*, the gadget G_i can be partitioned into 9 K_3's and 4 P_4's (see Figure 1.5(b));
- in case *(ii)*, the gadget $G_i - \{x, y, z\}$ can be partitioned into 8 K_3's and 4 P_4's (see Figure 1.5(c)).

It is easy to see that such a partition uses the maximum number of edge-disjoint triangles in the tripartite graph R, and only K_3's and P_4's are involved. By Lemma 1.2, this partition is an optimal solution of Ring Traffic Grooming for $C = 1$ in R. Let OPT be the number of vertices of such an optimal solution in R, and let t^* be the number of triangles

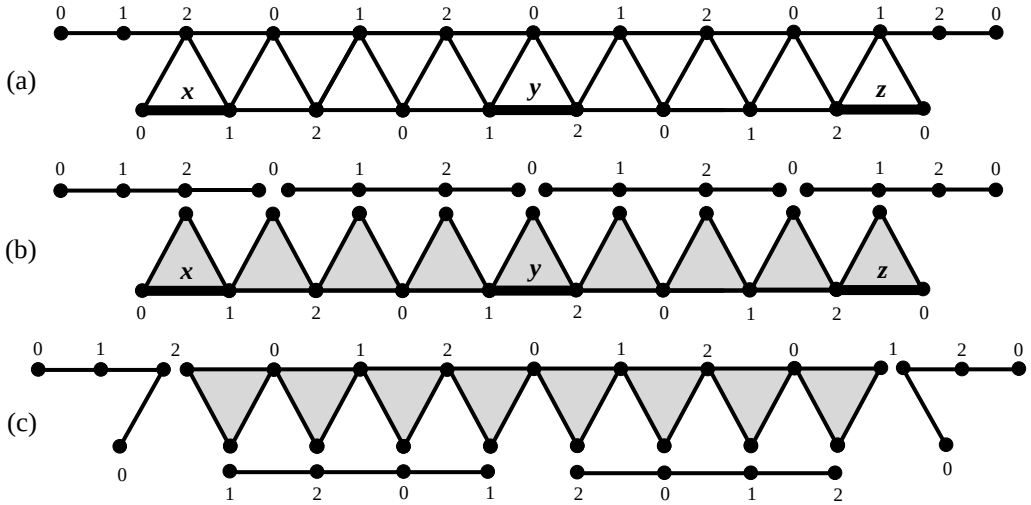

Figure 1.5: Request graphs used in the proof of Theorem 1.3: (a) gadget G_i corresponding to the set $c_i = \{x, y, z\}$. The labels of the vertices indicate the tripartition; (b) partition into 9 K_3's and 4 P_4's *with* the edges x, y, z ; (c) partition into 8 K_3's and 4 P_4's *without* the edges x, y, z.

in an optimal solution in R. (We simply write OPT and t^* instead of $OPT(R)$ and $t^*(R)$, respectively.) It is clear that

$$|E(R)| \leq OPT \leq 2|E(R)|. \tag{1.2}$$

We have seen in the proof of Lemma 1.2 that the cost A_t of any solution using t triangles satisfies $A_t \geq \frac{4}{3}|E(R)| - 3t$. We can also write

$$OPT = \frac{4}{3}|E(R)| - 3t^*. \tag{1.3}$$

Since MECT-B is Apx-hard in $\mathcal{R}$, there exists a constant $\varepsilon_0 > 0$ such that, unless P = NP, one cannot find in polynomial time more than $(1 - \varepsilon_0)t^*$ triangles in an arbitrary graph $R \in \mathcal{R}$. Therefore, the cost A of any solution of Ring Traffic Grooming that can be found in polynomial time satisfies

$$A \geq \frac{4}{3}|E(R)| - 3(1 - \varepsilon_0)t^* = OPT + 3\varepsilon_0 t^*, \tag{1.4}$$

where we have used Equation (1.3). On the other hand, from Equation (1.3) and using Equation (1.2) twice we get

$$t^* = \frac{4}{9}|E(R)| - \frac{OPT}{3} \geq \frac{4}{9}|E(R)| - \frac{|E(R)|}{3} = \frac{|E(R)|}{9} \geq \frac{OPT}{18}. \tag{1.5}$$

Combining Equations (1.4) and (1.5) yields that the cost A of any solution satisfies

$$A \geq OPT + 3\varepsilon_0 \frac{OPT}{18} = \left(1 + \frac{\varepsilon_0}{6}\right) OPT = (1 + \varepsilon_1)OPT,$$

with $\varepsilon_1 = \varepsilon_0/6 > 0$. Therefore, unless P = NP, Ring Traffic Grooming does not admit a PTAS for fixed $C = 1$. □

As expected, the result can be generalized to any $C \geq 1$.

Theorem 1.4 Ring Traffic Grooming *is* Apx*-complete for any fixed* $C \geq 1$*, even if the request graph has degree bounded by a constant* $B \geq 10$*. Thus, it does not admit a* PTAS *unless* P = NP*.*

Proof: The case $C = 1$ has been proved in Theorem 1.3, so assume henceforth that $C > 1$. The problem is in Apx for any $C \geq 1$ by Lemma 1.1. To prove the Apx-hardness, consider a $(2C + 1)$-partite graph as request graph, in such way that each cycle makes at least C tours around the center of the ring. At this point we can reduce the grooming problem to the problem of finding a maximum number of cycles of length $2C + 1$ in this graph (as in the case $C = 1$). This later problem is also Apx-complete by Theorem 1.2. The details follow.

Let G be a graph satisfying the conditions of Theorem 1.2: G is a $(2C + 1)$-partite graph, consisting of $2C + 1$ parts $A_0, \ldots, A_{2C}$ such that the only edges are between A_i and $A_{i+1 \pmod{2C+1}}$, $i = 0, \ldots, 2C$, and such that the graph induced between two consecutive parts of G forms a graph of girth at least $C + 1$. In order to simplify the presentation, suppose that this graph can be partitioned into C_{2C+1}'s.

Let $c_0, \ldots, c_{2C}$ be a permutation of the vertices of the cycle C_{2C+1}, such that the polygon $(c_0, \ldots, c_{2C})$ makes C tours around the center (for $C = 1$ take the triangle; for C arbitrary, let $c_i = \exp(\frac{2iC\pi}{2C+1})$). Now replace each vertex c_i with an interval consisting of vertices of A_i. In this cyclic representation of the graph G, each cycle makes at least C tours around the origin. To see this, recall that the only possible edges are between A_i and $A_{i+1 \pmod{2C+1}}$, $i = 0, \ldots, 2C$, and also that the graph induced between two consecutive parts has girth at least $C + 1$. This implies that every cycle should intersect each A_i at least once, and so this cycle makes at least C tours around the origin, as the original cycle $\{c_0, \ldots, c_{2C}\}$ does so.

Each cycle used in the solution should be of length exactly $2C + 1$, there is no cycle of smaller length, and longer cycles use each edge more than C times, as they make more than C tours around the origin. Then the problem is reduced to finding edge-disjoint cycles of length $2C + 1$, which is Apx-hard by Theorem 1.2. The proof of Theorem 1.3 can now be reproduced to obtain the same result for any C, replacing the factor $\frac{4}{3}$ for $C = 1$ (because the path with greatest density in any solution for $C = 1$ is a P_4) with a factor $\frac{2C+2}{2C+1}$ for a general C (because the path with greatest density in any solution for general C is P_{2C+2}). Hence, Ring Traffic Grooming is Apx-complete even for bounded number of requests per node $B \geq 10$. □

These ideas can be naturally extended to prove the Apx-completeness of Path Traffic Grooming for any fixed $C \geq 2$.

Theorem 1.5 Path Traffic Grooming *is* Apx*-complete for any fixed* $C \geq 2$*. Thus, it does not admit a* PTAS *unless* P = NP*.*

Proof: Again, the result holds even for bounded number B of requests per node, $B \geq 10$. We prove the result for $C = 2$, proceeding for $C > 2$ as in the proof of Theorem 1.4. Consider the family of request graphs $\mathcal{R}$ defined in the proof of Theorem 1.3, and place the three partition classes consecutively on the path one after another, as shown in Figure 1.4(b). Since each triangle induces load 2, minimizing the number of ADMs corresponds to finding the maximum number of edge-disjoint triangles. Therefore, the problem does not admit a PTAS unless P = NP. □

1.4 Approximating Traffic Grooming

We are now interested in finding good approximation algorithms considering C as part of the input. As discussed in Section 1.3, obtaining a $O(\sqrt{C})$-approximation is trivial. Since in practical applications SONET WDM rings are widely used as backbone optical networks [106, 170], the grooming factor is usually greater than the size of the network, i.e., $C \geq n$. Thus, it turns out to be important to find approximation algorithms with an approximation ratio not depending on C. A general approximation algorithm with this property is the main result of this section. It provides in the worst case a $O(n^{1/3} \log^2 n)$-approximation. We describe it for the ring topology, but exactly the same arguments provide an algorithm for the path. The main idea is to greedily find subgraphs with high density using approximation algorithms for the Dense k-Subgraph problem, which is defined as follows: given a graph G and an integer k, find an induced subgraph $H \subseteq G$ on k vertices with the greatest density among all subgraphs on k vertices. In [114] the authors provide a polynomial-time algorithm with approximation ratio $2n^{1/3}$. To simplify the presentation, suppose that $n = 2^t$ for some $t > 0$ (otherwise, introduce dummy vertices on the ring until getting size $n' = 2^t$, with $n' < 2n$):

Algorithm $\mathcal{A}$:

(1) Divide the request set into $\log n$ classes, such that in each class C_i the length of the requests lies in the interval $[2^i, 2^{i+1})$, $i = 0, \ldots, \log n - 1$. For each class C_i, the ring can be divided into intervals of length 2^i such that the only requests are between consecutive intervals. In this way we obtain $\frac{n}{2^i}$ subproblems for each class: each one consists in finding an optimal solution in a bipartite graph of size $2 \cdot 2^i$. More precisely, each subproblem can be formulated as:

Bipartite Traffic Grooming
Input: A bipartite graph R, and a grooming factor C.
Output: Partition of the edges of R into subgraphs R_λ such that $\lvert E(R_\lambda)\rvert \leq C$, for $1 \leq \lambda \leq \Lambda$.
Objective: Minimize $\sum_{\lambda=1}^{\Lambda} \lvert V(R_\lambda)\rvert$.

Find a solution to each Bipartite Traffic Grooming subproblem independently using step **(2)**, and output the union of all solutions.

(2) To find a solution to each Bipartite Traffic Grooming subproblem in a bipartite graph R, proceed greedily (until all edges are covered) by finding at step j a subgraph R_j of $G \setminus (R_1 \cup \cdots \cup R_{j-1})$ with at most C edges in the following way:
For each $k = 2, \ldots, 2C$ find a subgraph B_k of $R \setminus (R_1 \cup \cdots \cup R_{j-1})$ using the algorithm of [114] for the Dense k-Subgraph problem.
- If for some k^*, $|E(B_{k^*})| > C$, and $|E(B_j)| \leq C$ for all $j < k^*$, remove $|E(B_{k^*})| - C$ arbitrary edges from B_{k^*} and output the densest graph among $B_2, \ldots, B_{k^*-1}, B_{k^*}$.
- Otherwise, output the densest graph among $B_2, \ldots, B_{2C}$.

Let OPT be an optimal solution of Ring Traffic Grooming, and let OPT_1 be the cost of the solution obtained by solving *optimally* all the subproblems generated by step **(1)** of Algorithm $\mathcal{A}$. We prove a lemma before stating Theorem 1.6.

Lemma 1.3 *Let β be a given positive real number. Suppose that there exists an algorithm that finds in any bipartite graph R on at most n vertices, a subgraph with at most C edges which has density at least $1/\beta$ times the density of the densest subgraph with at most C edges. Then in the greedy procedure of step **(2)** of Algorithm $\mathcal{A}$, one obtains a solution of cost OPT_2 such that $OPT_2 \leq O(\log n) \cdot \beta \cdot OPT_1$.*

Proof: Let m be the number of edges of the request graph R, and let $R_1, R_2, \ldots, R_r$ be the subgraphs generated, in this order, by the above algorithm. We will prove that $\sum |V(R_i)| \leq \log(m) \cdot \beta \cdot OPT_1$. To prove this, we first enumerate the edges of R in order of appearance in R_i's: all the edges in R_1 will be enumerated $e_1, \ldots, e_{C_1}$ ($C_1 = |E(R_1)| \leq C$), all the edges in R_2 will be enumerated $e_{C_1+1}, \ldots, e_{C_1+C_2}$ ($C_2 = |E(R_2)| \leq C$), and so on. Let ρ_i be the density of the subgraph R_i, i.e., $\rho_i = \frac{|E(R_i)|}{|V(R_i)|}$, and $\Sigma = \sum |V(R_i)|$ the total cost of the solution. For every edge $e_j \in R_i$, we define $c(e_j) = \frac{1}{\rho_i}$. We claim that $\sum_j c(e_j) = \Sigma$. To prove this equality just note that $\sum_{e_j \in E(R_i)} c(e_j) = \frac{|E(R_i)|}{\rho_i} = |V(R_i)|$, and so $\sum_j c(e_j) = \sum_i |V(R_i)| = \Sigma$. Let us define R'_i to be the union of $R_i, R_{i+1}, \ldots, R_r$. We define ρ'_i to be the density of the densest subgraph of R'_i containing at most C edges. Let us take an optimal solution for R'_i, i.e., a decomposition of R'_i into subgraphs $A_1, \ldots, A_s$ such that $\sum_{k=1}^{s} |V(A_k)|$ is minimum. Let $\overline{\rho}_1, \ldots, \overline{\rho}_s$ be the density of these subgraphs. We have:

- $\forall k \leq s$, $\overline{\rho}_k \leq \rho'_i$: because each A_k is a subgraph of R'_i containing at most C edges, and ρ'_i is the density of the densest subgraph with at most C edges in R'_i.
- $\rho'_i \leq \beta\rho_i$: because we suppose that we can find an approximation of ρ'_i up to a factor $1/\beta$.

This implies that $\frac{1}{\overline{\rho}_k} \geq \frac{1}{\beta\rho_i}$, and so

$$\sum_k |V(A_k)| = \sum_k \frac{|E(A_k)|}{\overline{\rho}_k} \geq \sum_k \frac{|E(A_k)|}{\beta\rho_i} = \frac{|E(R'_i)|}{\beta\rho_i}.$$

But an optimal solution for R provides a solution for R'_i of cost at least the optimal solution for R'_i, i.e., $\sum_k |V(A_k)| \leq OPT_1$. Using this in the above inequality we get $\frac{1}{\rho_i} \leq \frac{\beta \cdot OPT_1}{|E(R'_i)|}$, and so for an edge $e_j \in R_i$ we have $c(e_j) = \frac{1}{\rho_i} \leq \frac{\beta \cdot OPT_1}{|E(R'_i)|} \leq \frac{\beta \cdot OPT_1}{m-j+1}$, and this proves that

$$\Sigma = \sum_j c(e_j) \leq \beta \cdot (\sum_j \frac{1}{m-j+1}) \cdot OPT_1 \leq \beta \cdot \log(m) \cdot OPT_1 \leq 2\beta \cdot \log(n) \cdot OPT_1.$$

□

Theorem 1.6 *$\mathcal{A}$ is a poly-time approximation algorithm that approximates* Ring Traffic Grooming *within a factor $O(n^{1/3} \log^2 n)$ for any $C \geq 1$.*

Proof: Algorithm $\mathcal{A}$ returns a valid solution of Ring Traffic Grooming, because each request is contained in some bipartite graph, and no request is counted twice. The runtime is polynomial in both n and C, because we run at most $2C-1$ times the algorithm of [114] for each subproblem, and there are $n(\sum_{i=0}^{t-1} \frac{1}{2^i}) - 1 = 2n - 3$ subproblems. We prove the approximation guarantee:

- We claim that $OPT_1 \leq 2 \log n \cdot OPT$. Indeed, let $\overline{c}_i$ be the optimal cost of the subset of requests of length in the interval $[2^i, 2^{i+1})$, $i = 0, \ldots, \log(n) - 1$. It is clear that $\overline{c}_i \leq OPT$ for each i , and thus $\sum_{i=0}^{\log n - 1} \overline{c}_i \leq \log n \cdot OPT$. Finally, $OPT_1 \leq 2 \sum_{i=0}^{\log n - 1} \overline{c}_i$, because each vertex is taken into account in two subproblems.

- The greedy procedure described in step **(2)** of Algorithm $\mathcal{A}$ outputs a graph whose density is at least $\frac{1}{2n^{1/3}}$ times the greatest density (with at most C edges) of the updated request graph. To see that, note that the optimal density is achieved by a subgraph on at most $2C$ vertices (it would be the case of C disjoint edges). Then, for each value of k, the algorithm of [114] finds a $2n^{1/3}$-approximation of the maximum number of edges of an induced subgraph on k vertices[1]. Thus, if we take the densest subgraph among $B_2, \ldots, B_{2C}$ (removing edges if necessary) we also obtain a $2n^{1/3}$-approximation of the greatest density of a subgraph with at most C edges. Let ρ_k be the density of B_k before removing edges. The explicit formula of the greatest density ρ that we output in step **(2)** of Algorithm $\mathcal{A}$ is:

  $$\rho = \max_{k \in \{2, \ldots, 2C\}} \min\left(\rho_k, \frac{C}{k}\right).$$

  The above formula justifies that the algorithm stops the search at $k = k^*$. Summarizing, we can use $\beta = 2n^{1/3}$ in Lemma 1.3.

- By combining the remarks above and Lemma 1.3 we obtain that the cost A returned by Algorithm $\mathcal{A}$ satisfies $A \leq 2n^{1/3} \cdot OPT_2 \leq 4n^{1/3} \log n \cdot OPT_1 \leq 8n^{1/3} \log^2 n \cdot OPT$.

□

[1] In fact, the improved approximation ratio of the Dense k-Subgraph problem is $O(n^\delta)$ for some constant $\delta < 1/3$ [114]. Obviously, the same applies to our algorithm, replacing the exponent 1/3 with the same $\delta < 1/3$.

We can improve the approximation ratio of the algorithm if all the requests have short length compared to the length of the ring. This situation is usual in practical applications since nodes may want to communicate only with their nearest neighbors. Let $f(n)$ be any function of n. If all the requests have length at most $f(n)$, then the above algorithm provides an approximation ratio of $O(f(n)^{1/3}\log^2 n)$. Indeed, in step **(2)** of Algorithm $\mathcal{A}$, we have to find dense subgraphs in bipartite graphs of size at most $2f(n)$, hence the factor $2n^{1/3}$ can be replaced with $2(2f(n))^{1/3}$.

Remark that all the instances of Dense k-Subgraph problem in our algorithm are bipartite. Using the results of [198], it is possible to obtain a better approximation ratio when the request graph is bipartite and satisfies some uniform density conditions.

Corollary 1.1 *If the request graph R is such that in any large enough subgraph $H \subseteq R$, a densest subgraph $(A \cup B, E)$ satisfies $|A|, |B| = O(\sqrt{C})$ and $|E| = \Omega(C)$, then for any constant $\varepsilon > 0$ there exists a polynomial-time algorithm for* Ring Traffic Grooming *with approximation ratio $O(n^{\varepsilon}\log^2 n)$.*

To end this section, it is interesting to mention that the results of [92] show that the density can be approximated within a constant factor two in the class of graphs excluding a fixed graph H as minor. Thus, if the request graph R is H-minor free (for instance if R is planar or of bounded genus), Algorithm $\mathcal{A}$ achieves an approximation factor of $O(\log^2 n)$.

1.5 Conclusions

The contribution of this chapter can be divided into two main parts: on the one hand, we stated inapproximability results for Ring Traffic Grooming and Path Traffic Grooming for fixed values of C. More precisely, we proved that Ring Traffic Grooming is Apx-complete for fixed $C \geq 1$, and that Path Traffic Grooming is Apx-complete for fixed $C \geq 2$. In other words, we ruled out the existence of a PTAS for fixed values of C. To prove this results we reduced Ring Traffic Grooming for $C = 1$ to the problem of finding the maximum number of edge-disjoint triangles in a graph of degree bounded by B (MECT-B for short). We proved that MECT-B is Apx-complete, and we generalized this reduction for Path Traffic Grooming and for all values of $C \geq 1$. On the other hand, we provided a polynomial-time approximation algorithm for Ring and Path Traffic Grooming with an approximation ratio not depending on C, considering C as part of the input.

A number of interesting questions remain open. First, when C is not part of the input, the non-existence of a PTAS blows the whistle to start the race of finding the best constant factor approximation for each value of C, for both the ring ($C \geq 1$) and the path ($C \geq 2$). We did not focus on this issue in this chapter.

Secondly, when C is part of the input, it is a challenging open problem to close the complexity gap of Traffic Grooming, that is, to provide an approximation algorithm with an approximation ratio matching the corresponding inapproximability result. We are convinced that the inherent difficulty of the problem resides in finding dense subgraphs with bounded number of edges. This problem is strongly related to the problem of finding

the densest subgraph with bounded number of vertices, which has been recently proved to have, essentially, the same difficulty as the DENSE k-SUBGRAPH problem [37]. The non-existence of a PTAS for the DENSE k-SUBGRAPH problem has been proved in [159] involving very technical proofs, and this is the best existing hardness result. A long-standing conjecture claims that there exists some constant $\varepsilon > 0$ such that finding a n^{ε}-approximation for DENSE k-SUBGRAPH is NP-hard [114]. As we proved in Section 1.4, an α-approximation for DENSE k-SUBGRAPH yields a $O(\alpha \log^2 n)$-approximation for RING TRAFFIC GROOMING. We suspect that a similar result in the other direction should also exist. Because of this, we conjecture that:

Conjecture 1.1 *There exists some constant $\delta > 0$, such that* RING TRAFFIC GROOMING *is* NP*-hard to approximate in polynomial time within a factor $O(n^{\delta})$ when the grooming factor C is part of the input.*

In [74, Proposition 2] it is shown that RING TRAFFIC GROOMING is in P for fixed n. Using terminology from parameterized complexity (see Section I.2.3), this result only shows that RING TRAFFIC GROOMING is in XP and not necessarily FPT (if n is the parameter). In [74] it is said that Michael Fellows has shown that if the number of ADMs is taken to be the parameter, then RING TRAFFIC GROOMING is in FPT. Unfortunately, the number of ADMs tends to be much larger than the ring size, so it remains an interesting open problem whether ring grooming is FPT if n is the parameter and C is part of the input.

Chapter 2

Bounded-degree Request Graph

In this chapter we consider the unidirectional ring topology with a generic grooming factor C. We introduce the pseudo-dynamic case where the request graph has bounded degree Δ, and our aim is to design a network (namely, place the ADMs at each node) being able to support *any* request graph with maximum degree at most Δ. The existing theoretical models in the literature are much more rigid, and do not allow such adaptability. We show that the problem is essentially equivalent to finding the least integer $M(C, \Delta)$ such that the edges of any graph with maximum degree at most Δ can be partitioned into subgraphs with at most C edges and each vertex appears in at most $M(C, \Delta)$ subgraphs. We establish the value of $M(C, \Delta)$ for almost all values of C and Δ, leaving open only the case where $\Delta \geq 5$ is odd, $\Delta \pmod{2C}$ is between 3 and $C - 1$, $C \geq 4$, and the request graph does not contain a perfect matching. For these open cases, we provide upper bounds that differ from the optimal value by at most one.

Keywords: optical networks, SONET over WDM, traffic grooming, ADM, graph decomposition, cubic graph, perfect matching, bridgeless graph.

2.1 Introduction

In this chapter we consider unidirectional SONET/WDM ring networks with symmetric requests. As already mentioned in Section II.2 (page 31), in this case the routing is unique and to each request between two nodes, we assign a wavelength and some bandwidth on this wavelength. If the traffic is uniform and any given wavelength can carry at most C requests, we can assign at most $\frac{1}{C}$ of the bandwidth to each request, C being the grooming factor. Furthermore, if the traffic requirement is symmetric, we may assume that symmetric requests are assigned the same wavelength, as it is easy to show (by exchanging wavelengths) that there exists an optimal solution where all symmetric requests are given

the same wavelength. Then each pair of symmetric requests uses $\frac{1}{C}$ of the bandwidth in the whole ring. If the two end-nodes are u and v, we need one ADM at node u and one at node v. The main point is that if two requests have a common end-node, they can share an ADM if they are assigned the same wavelength.

The traffic grooming problem for a unidirectional SONET ring with n nodes, grooming ratio C, and a symmetric request graph R has been modeled as a graph partition problem as follows (see [48,138]). Each edge of R corresponds to a pair of symmetric requests, and edges are colored by their assigned wavelength λ. All edges of color λ induce a connected subgraph B_λ of R, where each node corresponds to an ADM. The grooming constraint, i.e., the fact that a wavelength can carry at most C requests, translates to an upper bound C on the number of edges in each B_λ. The cost corresponds to the total number of vertices used in the subgraphs, and the objective is therefore to minimize $\sum_\lambda |V(B_\lambda)|$.

While most of previous work has focused on the case where the requests are given as input [J1,48,49,106,118,120,138,170], we consider the case where only the network topology is given, together with a bound Δ on the request graph. We would like to place, for each value of the grooming factor C, a minimum number of ADMs at each node in such a way that they could support *any* traffic pattern where each node is the end-node of at most Δ requests. This model is interesting because the network can support dynamic traffic without replacement of the ADMs.

From a practical point of view, it is interesting to design a network being able to support any request graph with maximum degree not exceeding a given constant. This situation is usual in real optical networks, since due to technology constraints the number of allowed communications for each node is usually bounded. This flexibility can also be thought from another point of view: if we have a limited number of available ADMs to place at the nodes of the network, then it is interesting to know which is the maximum degree of a request graph that our network is able to support, depending on the grooming factor. Equivalently, given a maximum degree and a number of available ADMs, it is useful to know which values of the grooming factor the network will support.

The aim of this chapter is to provide a theoretical framework to design such networks with dynamically changing traffic. We study the case where the physical network is given by an unidirectional ring, which is a widely used topology (for instance, in SONET rings). The problem we study can be formulated as a graph partitioning problem as follows.

Δ-Degree-Bounded Traffic Grooming in Unidirectional Rings
Input: Three integers n (size of the ring), C (grooming factor), and Δ (maximum degree).
Output: An assignment of $A(v)$ ADMs to each vertex v of the ring, in such a way that *for any request graph* G with maximum degree at most Δ, there exists a partition of $E(G)$ into subgraphs $\{B_\lambda\}_{1\leq\lambda\leq\Lambda} = \mathcal{B}$, such that:
(i) $|E(B_\lambda)| \leq C$ for all λ; and
(ii) each vertex $v \in V(G)$ appears in at most $A(v)$ subgraphs.
Objective: Minimize $\sum_v A(v)$.

The optimum to the above problem for each n, C, Δ is denoted by $A(n, C, \Delta)$. Before getting into details, we first fix the notation to be used in this chapter.

Notation. The (multi)graphs considered in this chapter are finite and without self-loops. A Δ-*graph* is a (multi)graph with maximum degree at most Δ. $\mathcal{G}_\Delta$ denotes the class of all Δ-graphs. A Δ-*regular* (multi)graph is a graph in which all vertices have degree Δ. An *almost* Δ-*regular* (multi)graph is a (multi)graph in which all vertices have degree Δ except possibly one which has degree $\Delta-1$. A *bridge* in a (multi)graph G is an edge whose removal disconnects G. A *matching* in a (multi)graph $G = (V, E)$ is a subset $M \subseteq E$ which contains each vertex at most once. A *perfect matching* is a matching containing all vertices. A *digon* is a cycle of length 2. A *trail* in a (multi)graph is a sequence $\{\{x_1, x_2\}, \{x_2, x_3\}, \ldots, \{x_{k-1}, x_k\}\}$ of distinct edges in which the second end of an edge is the first end of the next edge (the same pair of vertices may appear more than once if there is more than one edge between them). Vertices $x_2, x_3, \ldots, x_{k-1}$ of a trail are called *midpoints*. The *length* of a trail is the number of edges in it. Given a (multi)graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$, we denote by $G - V'$ the (multi)graph obtained from G by removing the vertices in V', the edges incident with vertices in V', and isolated vertices (if any). Similarly, given a subset of edges $E' \subseteq E$, we denote by $G - E'$ the (multi)graph obtained from G by removing the edges in E' and isolated vertices (if any). Given a graph with maximum degree at most Δ, a partition of G into subgraphs with at most C edges is called a C-*edge-partition* of G.

The function $A(n, C, \Delta)$ satisfies some straightforward properties.

Lemma 2.1 *The following statements hold:*

(i) $A(n, C, 1) = n$.

(ii) $A(n, 1, \Delta) = \Delta \cdot n$.

(iii) *If* $C' \geq C$, *then* $A(n, C', \Delta) \leq A(n, C, \Delta)$.

(iv) *If* $\Delta' \geq \Delta$, *then* $A(n, C, \Delta') \geq A(n, C, \Delta)$.

(v) $A(n, C, \Delta) \geq n$ *for all* $\Delta \geq 1$.

(vi) *If* $C \geq \frac{n\Delta}{2}$, $A(n, C, \Delta) = n$.

Proof:

(i) The request graph can consist in a perfect matching, so any solution uses 1 ADM per node.

(ii) A Δ-regular graph can be partitioned into $\frac{n\Delta}{2}$ disjoint edges.

(iii) Any solution for C is also a solution for C'.

(iv) If $\Delta' \geq \Delta$, the subgraphs with maximum degree at most Δ are a subclass of the class of graphs with maximum degree at most Δ'.

(v) Combine *(i)* and *(iv)*.

(vi) In this case all the edges of the request graph fit into one subgraph.

□

Organization of the chapter. In Section 2.2 we show that the Δ-Degree-Bounded Traffic Grooming in Unidirectional Rings problem is essentially equivalent to establishing the value of the parameter $M(C, \Delta)$ (see Definition 2.1) for each value of C and Δ. We solve the cases where $\Delta \geq 2$ is even in Section 2.3. In Section 2.4 we focus on the cases where $\Delta \geq 3$ is odd, leaving open only the cases where $\Delta \geq 5$ is odd, $\Delta \pmod{2C}$ is between 3 and $C-1$, $C \geq 4$, and the graph does not contain a perfect matching (see Table 2.1). In Section 2.4.6 we present an attempt to solve these remaining cases, that may lead to an eventual proof. Finally, Section 2.5 concludes the chapter.

2.2 The Parameter $M(C, \Delta)$

The following definition will play a fundamental role in the remainder of this chapter.

Definition 2.1 *Let $M(C, \Delta)$ be the smallest number M such that $A(n, C, \Delta) \leq M \cdot n$ for all $n \geq 1$.*

Lemma 2.2 *$M(C, \Delta)$ is a natural number.*

Proof: We know by Lemma 2.1 that, for any $C \geq 1$, $n \leq A(n, C, \Delta) \leq A(n, 1, \Delta) = \Delta \cdot n$. Suppose that M is not a natural number. That is, suppose that $r < M < r + 1$ for some positive integer r. Therefore, there must be at least $(r + 1 - M)n$ vertices with at most r ADMs each. For each n, let $V_{n,r}$ be the subset of vertices of the request graph with at most r ADMs. Then, since $r+1-M > 0$, we have that $\lim_{n\to\infty} |V_{n,r}| = \infty$. In other words, there is an arbitrarily big subset of vertices with at most r ADMs per vertex. But we can consider a request graph with maximum degree at most Δ on the set of vertices $V_{n,r}$, and this means that with r ADMs per node we can construct a C-edge-partition, a contradiction with the optimality of M. □

If the request graph is further restricted to belong to a subclass of graphs $\mathcal{C} \subseteq \mathcal{G}_\Delta$, then the corresponding positive integer is denoted by $M(C, \Delta, \mathcal{C})$.

By the discussion above, $A(n, C, \Delta)$ is of the form $A(n, C, \Delta) = M(C, \Delta) \cdot n - \alpha(C, \Delta)$, where $M(C, \Delta)$ and $\alpha(C, \Delta)$ are integers depending only on C and Δ. Suppose that a Δ-graph H requires at least $M(C, \Delta) + 1$ ADMs at some vertex. Since any Δ-graph must be supported with the same ADMs, by relabeling the vertices of H we could force at least $M(C, \Delta) + 1$ ADMs in $\Omega(n)$ nodes of the network. This would contradict the definition of $M(C, \Delta)$. Therefore, each vertex can appear in at most $M(C, \Delta)$ subgraphs. So we may conclude the following.

Remark 2.1 *For each value of C and Δ, Δ-Degree-Bounded Traffic Grooming in Unidirectional Rings reduces to finding the least integer $M(C, \Delta)$ such that the edges of any Δ-graph can be partitioned into subgraphs with at most C edges and each vertex appears in at most $M(C, \Delta)$ subgraphs.*

This allows us to give an equivalent definition of $M(C,\Delta)$. Let $G \in \mathcal{G}_\Delta$ and let $\mathcal{P}_C(G)$ be the set of C-edge-partitions of G. For $P \in \mathcal{P}_C(G)$, let $\mathrm{occ}(P)$ be the maximum number of occurrences of a vertex in the partition, that is,

$$\mathrm{occ}(P) = \max_{v \in V(G)} |\{B_\lambda \in P : v \in B_\lambda\}|, \quad \text{and then} \quad M(C,\Delta) = \max_{G \in \mathcal{G}_\Delta} \left(\min_{P \in \mathcal{P}_C(G)} \mathrm{occ}(P) \right).$$

In the remainder of this chapter, we use Remark 2.1 and focus on determining $M(C,\Delta)$ for each value of C and Δ. Observe also that any Δ-graph H is a subgraph of some Δ-regular graph G (with possibly more vertices). Note also that if we restrict a partition of G to the vertices of H, the number of occurrences of the vertices cannot increase. Therefore,

Remark 2.2 *$M(C,\Delta) = M(C,\Delta,\mathcal{C})$, where $\mathcal{C}$ is the class of Δ-regular graphs.*

The following lemma will be used throughout the chapter.

Lemma 2.3 *The following statements hold trivially from Lemma 2.1:*

(i) $M(C,1) = 1$ *for all* $C \geq 1$.

(ii) $M(1,\Delta) = \Delta$ *for all* $\Delta \geq 1$.

(iii) *If* $C' \geq C$, *then* $M(C',\Delta) \leq M(C,\Delta)$.

(iv) *If* $\Delta' \geq \Delta$, *then* $M(C,\Delta') \geq M(C,\Delta)$.

(v) $M(C,\Delta) \leq \Delta$ *for all* $C,\Delta \geq 1$.

The following proposition establishes a general lower bound on $M(C,\Delta)$, that will allow us to prove in many cases the optimality of the constructions of the next sections.

Proposition 2.1 $M(C,\Delta) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$ *for all values of* C,Δ.

Proof: Erdös and Sachs [113] proved that for any integer k there exist k-regular graphs with arbitrary large girth. For each value of C and Δ, let G be a Δ-regular graph with girth at least $C+1$, and let $n = |V(G)|$. Clearly all the subgraphs (with at most C edges) involved in the partition of the $\Delta n/2$ edges of G are trees. Therefore, the total number of vertices of any partition is at least $\frac{\Delta(C+1)}{2C} n$ (this can be easily seen using that the function $(x+1)/x$ with $1 \leq x \leq C$ is minimized when $x = C$). Then necessarily a vertex must occur in at least $\frac{\Delta(C+1)}{2C}$ subgraphs. By the definition of $M(C,\Delta)$, the lower bound follows. □

2.3 Case $\Delta \geq 2$ Even

In this section we establish the value of $M(C, \Delta)$ for $\Delta \geq 2$ even and any value of C.

Theorem 2.1 *Let $\Delta \geq 2$ be even. Then for any $C \geq 1$, $M(C, \Delta) = \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$.*

Proof: The lower bound follows from Proposition 2.1. Let us give an explicit construction for any Δ-regular graph $G = (V, E)$. Orient the edges of G in an Eulerian tour, and assign to each vertex $v \in V$ its $\Delta/2$ out-edges, namely E_v^+. For each $v \in V$, partition E_v^+ into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with C edges centered at v (except, possibly, one star with fewer edges). Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times. Therefore, the number of occurrences of each vertex in this partition is

$$\left\lceil \frac{\Delta}{2C} \right\rceil + \frac{\Delta}{2} = \left\lceil \frac{\Delta}{2} \left(1 + \frac{1}{C} \right) \right\rceil = \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil. \qquad \square$$

Note that for the special case $\Delta = 2$, Theorem 2.1 implies that $M(C, 2) = 2$ for all $C \geq 1$. In fact, for $\Delta = 2$ it is possible to give the exact expression of the cost function $A(n, C, 2)$. Indeed, it is easy to see that given a set of disjoint cycles, we can always find a C-edge-partition such that $C - 1$ prescribed (arbitrary) vertices appear in only one subgraph. On the other hand, if we pretend that at least C vertices appear in at most one subgraph, we can consider as request graph a cycle of length at least $C + 1$ containing the prescribed C vertices, and then necessarily one of those vertices appears in at least two subgraphs of any C-edge-partition, a contradiction. Summarizing, $A(n, C, 2) = 2n - (C - 1)$.

2.4 Case $\Delta \geq 3$ Odd

The cases where Δ is odd turn out to be inherently much more complicated than the cases where Δ is even. In Section 2.4.1, we present a general construction which differs from the lower bound of Proposition 2.1 by at most 1, and we determine when this construction is optimal. In Section 2.4.2 we provide an improved lower bound when $\Delta \equiv C \pmod{2C}$, which meets our upper bound. In Section 2.4.4 we solve the case $\Delta = 3$ and $C = 4$, which was the only unsolved case for $\Delta = 3$. In Section 2.4.5 we present an optimal construction for graphs with a perfect matching, after proving that the lower bound of Proposition 2.1 still holds when the request graph is restricted to have a perfect matching. We then discuss in Section 2.4.3 the relation of the parameter $M(C, \Delta)$ with the *linear C-arboricity* [34, 53, 201]. Finally, we describe in Section 2.4.6 an attempt to solve the remaining cases where $\Delta \geq 5$ is odd, using the ideas developed in the previous sections.

2.4.1 General upper bound

The following proposition provides a general upper bound, which differs from the lower bound of Proposition 2.1 by at most 1.

Proposition 2.2 *Let $\Delta \geq 3$ be odd. Then for any $C \geq 1$, $M(C, \Delta) \leq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} + \frac{C-1}{2C} \right\rceil$.*

Proof: Let G be a Δ-regular graph. Since Δ is odd, $|V(G)|$ is even. Add a perfect matching M to G to obtain a $(\Delta+1)$-regular multigraph G'. Orient the edges of G' in an Eulerian tour, and assign to each vertex $v \in V(G')$ its $(\Delta+1)/2$ out-edges E_v^+. Remove the edges of M and, as in the case Δ even, partition E_v^+ into stars with at most C edges. To count the number of occurrences of each vertex, we distinguish two cases. If an edge of M is in E_v^+, then v appears as center in $\left\lceil \frac{\Delta-1}{2C} \right\rceil$ stars and as a leaf in $\Delta - \frac{\Delta-1}{2}$ stars. Summing both terms yields

$$\left\lceil \frac{\Delta-1}{2C} \right\rceil + \Delta - \frac{\Delta-1}{2} = \left\lceil \frac{C+1}{C}\frac{\Delta}{2} + \frac{C-1}{2C} \right\rceil.$$

Otherwise, if no edge of M is in E_v^+, the number of occurrences of v is

$$\left\lceil \frac{\Delta+1}{2C} \right\rceil + \Delta - \frac{\Delta+1}{2} = \left\lceil \frac{C+1}{C}\frac{\Delta}{2} + \frac{1-C}{2C} \right\rceil \leq \left\lceil \frac{C+1}{C}\frac{\Delta}{2} + \frac{C-1}{2C} \right\rceil.$$

□

The upper bound of Proposition 2.2 and the lower bound of Proposition 2.1 are equal for, roughly speaking, half of the pairs C, Δ, as shown in the following corollary.

Corollary 2.1 *Let $\Delta \geq 3$ be odd. If $\Delta \pmod{2C} = 1$ or $\Delta \pmod{2C} \geq C+1$, then $M(C,\Delta) = \left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$.*

Proof: Let $\Delta = \lambda \cdot 2C + h$, with h odd, $1 \leq h \leq 2C-1$. Writing $k := \lambda(C+1) + \frac{h-1}{2}$, the lower bound of Proposition 2.1 equals $k + \left\lceil \frac{1}{2} + \frac{h}{2C} \right\rceil$, and the upper bound of Proposition 2.2 equals $k + \left\lceil 1 + \frac{h-1}{2C} \right\rceil$. If $h = 1$ both bounds equal $k+1$, and if $h \geq C+1$ both bounds equal $k+2$. □

In particular, when $C = 2$ and Δ is odd, $\Delta \pmod{2C}$ is either 1 or 3, and then by Corollary 2.1 the lower bound is attained, as stated in the following corollary.

Corollary 2.2 (Case $C = 2$) *For any $\Delta \geq 3$ odd, $M(2,\Delta) = \left\lceil \frac{3\Delta}{4} \right\rceil$.*

For all the cases we solved so far, the value of $M(C,\Delta)$ equals the lower bound of Proposition 2.1. It seems natural to think that the value $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$ may be always attained. We shall see in the next section that this is not true. Namely, we prove in Theorem 2.2 that if $\Delta \equiv C \pmod{2C}$, then $M(C,\Delta) = \left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil + 1$.

2.4.2 Improved lower bound

In this section we prove a new lower bound which strictly improves on Proposition 2.1 when $\Delta \equiv C \pmod{2C}$.

Theorem 2.2 *Let $\Delta \geq 3$ be odd and let $\Delta \equiv C \pmod{2C}$. Then $M(C,\Delta) = \left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil + 1$.*

Proof: We prove that if $\Delta = kC$ with k odd, then $M(C,\Delta) \geq \left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil + 1$ and thus, by Proposition 2.2, $M(C,\Delta)$ is equal to $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil + 1$. Since both Δ and k are odd, so is C, and therefore $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil = k \cdot \frac{C+1}{2}$.

We proceed to build a Δ-regular graph G with no C-edge-partition where each vertex is incident to at most $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$ subgraphs, hence implying that $M(C,\Delta) > \left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$. First, we construct a graph H where all vertices have degree Δ except one which has degree $\Delta - 1$. Furthermore, we build H so that it has girth strictly greater than C. H exists by [68,113]. Make Δ copies of H and add a cut-vertex v joined to all vertices of degree $\Delta - 1$ to make our Δ-regular graph G (see Figure 2.1 for an example of the construction of such a graph for $\Delta = C = 3$).

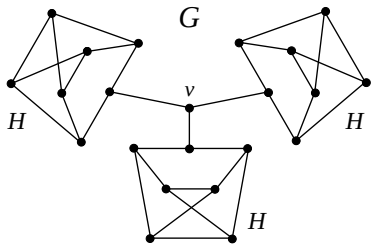

Figure 2.1: Cubic graph with girth 4, which is a counterexample showing that $M(3,3) = 3$.

Now suppose for the sake of contradiction that there is a C-edge-partition $\mathcal{B}$ of G where each vertex is incident to at most $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$ subgraphs. Since the girth of G is greater than C, all the subgraphs in $\mathcal{B}$ are trees. Since $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil < \Delta$, v must have degree at least 2 in some subgraph $T' \in \mathcal{B}$. Since $|E(T')| \leq C$, the tree T' contains at most $\left\lfloor \frac{C-2}{2} \right\rfloor = \frac{C-3}{2}$ edges of a copy H' of H intersecting T'. Now we only work in H'. Let $\alpha = |E(T' \cap H')| \leq \frac{C-3}{2}$ (note that $\alpha = 0$ for $C = \Delta = 3$).

Let $\mathcal{B}' = \{B \cap H'\}_{B \in (\mathcal{B} - \{T'\})}$, with the empty subgraphs removed. That is, $\mathcal{B}'$ contains the subgraphs in $\mathcal{B}$ that partition the edges in H' that are not in T'. Let $n = |V(H')|$, which is odd as in H' there is one vertex of degree $\Delta - 1$ and all the others have degree Δ. Therefore, the total number of edges of the trees in $\mathcal{B}'$ is

$$\sum_{T \in \mathcal{B}'} |E(T)| \;=\; |E(H')| - \alpha \;=\; \frac{n\Delta - 1}{2} - \alpha \;=\; \frac{nkC - 1}{2} - \alpha. \tag{2.1}$$

As $\alpha \leq \frac{C-3}{2}$, from Equation (2.1) we get

$$\sum_{T \in \mathcal{B}'} |E(T)| \;\geq\; \frac{nkC - 1}{2} - \frac{C-3}{2} \;=\; \left(\frac{nk-1}{2}\right) \cdot C + 1. \tag{2.2}$$

As each tree in $\mathcal{B}'$ has at most C edges, from Equation (2.2) we get that $|\mathcal{B}'|$, the number of trees in $\mathcal{B}'$, satisfies

$$|\mathcal{B}'| \;\geq\; \left\lceil \frac{nk-1}{2} + \frac{1}{C} \right\rceil \;=\; \frac{nk-1}{2} + \left\lceil \frac{1}{C} \right\rceil \;=\; \frac{nk-1}{2} + 1. \tag{2.3}$$

Clearly, the total number of vertices in the trees in $\mathcal{B}'$ is exactly the total number of edges in the trees in $\mathcal{B}'$ plus the number of trees in B', that is, $\sum_{T\in\mathcal{B}'}|V(T)| = \sum_{T\in\mathcal{B}'}|E(T)| + |\mathcal{B}'|$. On the other hand, the tree T' contains $\alpha+1$ vertices of H', that is, $|V(T'\cap H')| = \alpha+1$. Therefore, using Equations (2.1) and (2.3), we get that the total number of occurrences of the vertices in H' in some tree of $\mathcal{B}$ is

$$\begin{aligned}\sum_{v\in V(H')}|\{T\in\mathcal{B} : v\in T\}| &= \sum_{T\in\mathcal{B}'}|V(T)| + |V(T'\cap H')| = \sum_{T\in\mathcal{B}'}|E(T)| + |\mathcal{B}'| + \alpha + 1\\ &= \frac{nkC-1}{2} - \alpha + |\mathcal{B}'| + \alpha + 1 \geq \frac{nkC-1}{2} + \frac{nk-1}{2} + 1 + 1\\ &= nk\cdot\frac{C+1}{2} + 1 = n\cdot\left\lceil\frac{C+1}{C}\frac{\Delta}{2}\right\rceil + 1\ ,\end{aligned}$$

which implies that at least one vertex of H' appears in at least $\left\lceil\frac{C+1}{C}\frac{\Delta}{2}\right\rceil+1$ subgraphs, which is a contradiction to $\mathcal{B}$ being a C-edge-partition of G in which each vertex appears in at most $\left\lceil\frac{C+1}{C}\frac{\Delta}{2}\right\rceil$ subgraphs. The theorem follows. □

It turns out that Theorem 2.2 allows us to find the value of $M(3,\Delta)$ for any $\Delta\geq 3$ odd.

Corollary 2.3 (Case $C=3$) *For any $\Delta\geq 3$ odd, $M(3,\Delta) = \left\lceil\frac{2\Delta+1}{3}\right\rceil$.*

Proof: If $\Delta\equiv 1 \pmod 6$ or $\Delta\equiv 5 \pmod 6$, then by Corollary 2.1, $M(3,\Delta) = \left\lceil\frac{2\Delta}{3}\right\rceil = \left\lceil\frac{2\Delta+1}{3}\right\rceil$. Otherwise, if $\Delta\equiv 3 \pmod 6$, then by Theorem 2.2, $M(3,\Delta) = \left\lceil\frac{2\Delta}{3}\right\rceil + 1 = \left\lceil\frac{2\Delta+1}{3}\right\rceil$. □

2.4.3 Relation of $M(C,\Delta)$ with the linear C-arboricity

A result of Thomassen [201], which settled a conjecture of Bermond *et al.* [53], states that the edges of a cubic graph can be 2-colored such that each monochromatic component is a path of length at most 5. That is, in such a coloring (that can be seen as a partition into paths) each vertex appears in exactly 2 paths with at most 5 edges each. Therefore, combining this result with *(iii)* of Lemma 2.3 we deduce that $M(C,3) = 2$ for any $C\geq 5$.

Let us now discuss how these ideas can be extended to other values of C and Δ. A *linear C-forest* in a graph is a forest consisting of paths of length at most C. The *linear C-arboricity* of a graph G is the minimum number of linear C-forests required to partition $E(G)$, and is denoted by $la_C(G)$ [53]. Let $la_C(\Delta) = \max_{G\in\mathcal{G}_\Delta} la_C(G)$. Clearly $M(C,\Delta)\leq la_C(\Delta)$ for all C,Δ, since the paths in a *linear C-forest* are graphs with at most C edges. Therefore, the following upper bound given by Alon *et al.* [34] also applies to $M(C,\Delta)$.

Theorem 2.3 (Alon *et al.* [34]) *There is an absolute constant $\beta>0$ such that for $\sqrt{\Delta} > C\geq 2$,*

$$la_C(\Delta) \leq \frac{C+1}{C}\frac{\Delta}{2} + \beta\sqrt{C\Delta\log\Delta}. \tag{2.4}$$

It turns out that the first addend of the right-hand side of Equation (2.4) is equal to the lower bound of Proposition 2.1, so Theorem 2.3 provides an additive $O(\sqrt{C\Delta\log\Delta})$-approximation of $M(C,\Delta)$ for $\sqrt{\Delta} > C\geq 2$. Although we have improved this bound for $M(C,\Delta)$ in Sections 2.3 and 2.4.1, the relation between $M(C,\Delta)$ and $la_C(\Delta)$ is of theoretical interest by its own.

2.4.4 Case $\Delta = 3$, $C = 4$

As discussed in Section 2.4.3, $M(C,3) = 2$ for $C \geq 5$. On the other hand, Theorem 2.2 implies that $M(3,3) = 3$, so by *(ii)* and *(iii)* of Lemma 2.3 we have that $M(C,3) = 3$ for $C \leq 3$. Therefore, the interesting question is whether $M(4,3)$ equals 2 or 3. The remainder of this section is devoted to prove that $M(4,3) = 2$ (see Corollary 2.5). First we need a classical result concerning cubic graphs and an easy extension to cubic multigraphs.

Theorem 2.4 (Petersen [178]) *Any cubic bridgeless graph has a perfect matching.*

Corollary 2.4 *Any cubic bridgeless multigraph without self-loops has a perfect matching.*

Proof: Let G be a cubic multigraph without self-loops. We can assume that G has no triple edges, otherwise G has only 2 vertices and any of the 3 edges is a perfect matching. Consider the simple graph G' built from G as follows: for each digon $\{\{u,v\},\{u,v\}\}$, add 2 new vertices s_{uv} and t_{uv}, and replace the digon with the edges $\{u,s_{uv}\},\{u,t_{uv}\},\{v,s_{uv}\},\{v,t_{uv}\}$, and $\{s_{uv},t_{uv}\}$. By Theorem 2.4, G' has a perfect matching M'. We now construct a perfect matching M of G from M'. For each edge $e \in M'$ such that e was also an edge of G, put e in M'. For each digon $\{\{u,v\},\{u,v\}\}$ of G, if any of the pairs $\{\{u,s_{uv}\},\{v,t_{uv}\}\}$ or $\{\{u,t_{uv}\},\{v,s_{uv}\}\}$ is in M', put one of the copies of $\{u,v\}$ in M. Otherwise, $\{s_{uv},t_{uv}\}$ belongs to M' and we do nothing. It is easy to check that M is a perfect matching of G. □

We are ready to prove the main result of this section.

Theorem 2.5 *The edges of every almost 3-regular multigraph G without self-loops can be partitioned into a set $\mathcal{W} = \{W_1, W_2, \ldots, W_k\}$ of trails of length at most 4 such that each vertex appears as the midpoint of a trail.*

Proof: Suppose the theorem is false and let G be a counterexample with the minimum number of vertices. G is connected as otherwise, we can take the union of the partitions of its connected components, which exist by minimality of G.

Case 1: G contains a bridge $e = \{u,v\}$. Then $G - \{e\}$ has exactly two components: U containing u and V containing v. Without loss of generality, we may choose U to be the component with no degree 2 vertex in G and e is chosen so that U is maximal with this property. Thus this component U of $G - \{e\}$ is almost 3-regular (only u has degree 2). By minimality of G, U can be partitioned into a set $\mathcal{W}^u$ of trails as in the statement of the theorem.

If v has degree 2 in G then $V - \{v\}$ is almost 3-regular. By minimality of G, $V - \{v\}$ can be partitioned into a set $\mathcal{W}^v$ of trails as in the theorem. Now the only edges of G not in any trail in $\mathcal{W}^u \cup \mathcal{W}^v$ are those incident to v. Thus taking $\mathcal{W}^u \cup \mathcal{W}^v$ together with a trail consisting of the 2 edges incident to v (which has v as a midpoint) yields the required partition of the edges of G into trails. This contradicts the fact that G is a counterexample.

If v has degree 3 in G, let x, y be the neighbors of v in V (see Figure 2.2(a)). We can assume $x \neq y$ (i.e., $\{v, x\}$ and $\{v, y\}$ are not parallel edges) since otherwise, the third edge incident to $x = y$ is a cut edge whose choice (instead of e) would increase the size of U. Let H be the graph obtained from $V - \{v\}$ by adding an edge $f = \{x, y\}$ (see Figure 2.2(b)). By minimality of G, H can be partitioned into a set $\mathcal{W}^v$ of trails. We now attempt to transform $\mathcal{W}^u \cup \mathcal{W}^v$ into a partition of G into trails.

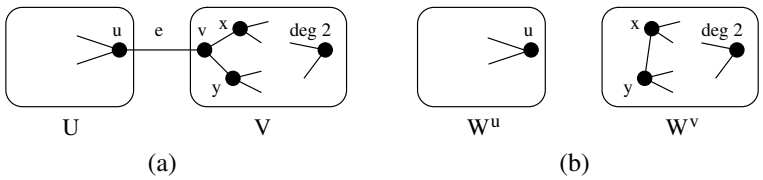

Figure 2.2: (a) A bridge $e = \{u, v\}$ in an almost 3-regular graph G with components U and V of $G - \{e\}$. (b) Graphs smaller than G from which we obtain a partition into trails W^u and W^v.

The edge f appears in some trail $\{W_1, \{x, y\}, W_2\}$ of $\mathcal{W}^v$, where W_1 is a (possibly empty) trail ending at x and W_2 is a (possibly empty) trail starting at y. At least one of the subtrails $\{W_1, \{x, y\}\}$ or $\{\{x, y\}, W_2\}$ has fewer than 3 edges. Without loss of generality, it is $\{W_1, \{x, y\}\}$. Replace this trail with $\{W_1, \{x, v\}, \{v, u\}\}$ which has length at most 4, and $\{\{v, y\}, W_2\}$ which has length less than or equal to $\{W_1, \{x, y\}, W_2\}$. Note that x and v are midpoints of the first trail and y is the midpoint of the second trail. Furthermore, any other vertex which was a midpoint in $\{W_1, \{x, y\}, W_2\}$ is still a midpoint (since W_1 and W_2 appear as subtrails).

Thus the union of $\mathcal{W}^u$ and $\mathcal{W}^v$ with the above replacement yields a partition of G into trails of length at most 4 with the desired property, which is a contradiction.

Case 2: G does not contains a bridge. If G is 3-regular, let $G' = G$. Otherwise, let G' be the graph obtained from G by replacing the vertex of degree 2 with an edge between its neighbors. Note that G' is 3-regular and contains no bridges. Therefore, by Corollary 2.4, G' contains a perfect matching $M \subseteq E(G')$.

Since G' is 3-regular, $G' - M$ is 2-regular. Thus, $G' - M$ is a union of disjoint cycles. We can orient the cycles of $G' - M$ so that each vertex v has exactly one edge e_v pointing towards v. For each edge $\{u, v\} \in M$, $W_{uv} = \{e_u, \{u, v\}, e_v\}$ is a trail of length 3 (see Figure 2.3). Note that $\mathcal{W} = \{W_{uv} \mid \{u, v\} \in M\}$ is a partition of the edges of G' into trails of length 3. Furthermore, every vertex u in the matching appears as the midpoint of the trail corresponding to the edge of the matching in which u appears. Since M is a perfect matching, every vertex appears as the midpoint of some trail in $\mathcal{W}$. Thus $G' \neq G$ as otherwise, we have constructed a partition as required by the theorem. So G has a vertex v of degree 2 which we replaced with an edge $e = \{x, y\}$ to obtain G'. Let $W = \{W_1, \{x, y\}, W_2\}$ be the trail in $\mathcal{W}$ containing e, and recall that W has length 3. Replacing W with $\{W_1, \{x, v\}, \{v, y\}, W_2\}$ in $\mathcal{W}$ yields a partition of $E(G)$ into trails of length at most 4, which is a contradiction. □

Note that the simple trees with some vertex of degree 3 and the digon with a pendant edge at each side are *not* allowed in the partition stated in Theorem 2.5, since these graphs cannot be thought of as trails. The following corollary settles the value of $M(4, 3)$.

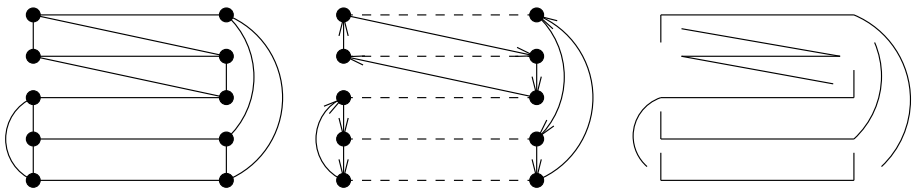

Figure 2.3: (a) A 3-regular graph G' with no bridges. (b) A matching M of G' (shown in dashed lines) and an orientation of the cycles of $G' - M$. (c) A partition of the edges of G' into trails of length 3 using M and the orientation of the cycle of $G' - M$ in (b).

Corollary 2.5 $M(4, 3) = 2$.

Proof: By Remark 2.2, we may restrict ourselves to 3-regular graphs. Thus, a 3-regular graph G is almost 3-regular and we may apply Theorem 2.5 to obtain a partition $\mathcal{W}$. Let $\mathcal{B} = \{E(W)\}_{W \in \mathcal{W}}$. Each vertex of G appears in at most two elements of $\mathcal{B}$, as G is 3-regular and each vertex appears as the midpoint of some trail in $\mathcal{W}$. □

2.4.5 Optimal construction for graphs with a perfect matching

In this section we focus on the case where the Δ-regular graphs are further restricted to contain a perfect matching. First observe that the proof of the general lower bound provided in Proposition 2.1 does not imply that the same lower bound carries over to Δ-regular graphs with a perfect matching. Indeed, the proof of Proposition 2.1 uses the existence of Δ-regular graphs with girth at least $C+1$, but those graphs may not necessarily contain a perfect matching. Fortunately, we can prove that the lower bound does not decrease when we assume that the graph contains a perfect matching.

Proposition 2.3 *Let $\Delta \geq 3$ be odd and let C be the class of Δ-regular graphs that contain a perfect matching. Then $M(C, \Delta, C) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$ for all $C \geq 1$.*

Proof: We shall construct a Δ-regular graph G with a perfect matching and girth at least $C + 1$, and then the proof of Proposition 2.1 applied to G yields the desired bound. The details follow.

For any two positive integers Δ and C, Chandran provided in [68, Section 2.1] an explicit and simple construction of a graph H such that

- H has girth strictly greater than C;
- H contains a perfect matching (in fact, H is obtained from a perfect matching by adding the appropriate edges); and
- The degree of the vertices of H is either $\Delta - 2$, $\Delta - 1$, or Δ.

We will construct from H our Δ-regular graph G. Let $v_1, \ldots, v_{k_1}$ be the vertices of degree $\Delta-1$ in H, and let $v_{k_1+1}, \ldots, v_{k_1+k_2}$ be the vertices of degree $\Delta-2$ in H. Let F be a (k_1+2k_2)-regular graph with girth at least $C+1$ (which exists by the result of Erdös and Sachs [113]), and let $f = |V(F)|$. Let the vertices of F be $u^1, \ldots, u^f$. To construct G, first make f copies of H, and let v_i^j be the copy of vertex v_i in the j-th copy of H, for $i = 1, \ldots, k_1 + k_2$ and $j = 1, \ldots, f$. Intuitively, each copy of H corresponds to a vertex of F. We now add $|E(F)|$ edges among the f copies of H as follows. We assign labels from 1 to $k_1 + 2k_2$ to the vertices of H with degree less than Δ in the following way: for $i = 1, \ldots, k_1$, vertex v_i gets label i, and for $i = k_1 + 1, \ldots, k_1 + k_2$, vertex v_i gets labels i and $k_2 + i$. For each vertex of F, we label arbitrarily the edges incident to it with distinct integers from 1 to $k_1 + 2k_2$ (recall that F is $(k_1 + 2k_2)$-regular). This way, each edge of F gets two labels, one from each end-vertex. Then, for each edge $\{u^{j_1}, u^{j_2}\} \in E(F)$ with labels (ℓ_1, ℓ_2), we add an edge between the vertices labeled ℓ_1 and ℓ_2 in the j_1-th and j_2-th copies of H, respectively.

This completes the construction of G. Note that the copies of the vertices that had degree Δ in H have also degree Δ in G. Since one (resp. two) edges have been added to each vertex of degree $\Delta - 1$ (resp. $\Delta - 2$), it is clear that G is Δ-regular. Since each copy of H had a perfect matching and no edge of any copy of H has been removed, G has also a perfect matching. Finally, the girth of G is at least $C + 1$. Indeed, the girth of each copy of H is at least $C + 1$ by [68]. Therefore, each cycle c of length at most C in G should visit strictly more than one copy of H. By the construction of G, such a cycle c in G would induce a cycle of length at most C in F among the vertices corresponding to the copies of H visited by c. But this is impossible as the girth of F is at least $C + 1$. □

We are now ready to provide an optimal construction for all $\Delta \geq 3$ odd and $C \geq 1$ when the request graph is restricted to have a perfect matching.

Proposition 2.4 *Let $\Delta \geq 3$ be odd and let $\mathcal{C}$ be the class of Δ-regular graphs that contain a perfect matching. Then $M(C, \Delta, \mathcal{C}) = \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$ for all $C \geq 1$.*

Proof: The lower bound follows from Proposition 2.3. To prove the upper bound, let G be a Δ-regular with a perfect matching M. Then $G - M$ is $(\Delta - 1)$-regular, with $\Delta - 1$ even. We orient the edges of $G - M$ in an Eulerian tour, and assign to each vertex $v \in V(G)$ its $\frac{\Delta-1}{2}$ out-edges E_v^+. We distinguish three cases.

(1) $\Delta < C$**.** For each edge $\{u, v\} \in M$, build a tree with Δ edges consisting of $\{u, v\}$, $\frac{\Delta-1}{2}$ edges from E_u^+, and $\frac{\Delta-1}{2}$ edges from E_v^+. The number of occurrences of each vertex is $1 + \Delta - \frac{\Delta+1}{2} = \frac{\Delta+1}{2}$. The lower bound equals $\left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil = \frac{\Delta-1}{2} + \left\lceil \frac{1}{2} + \frac{\Delta}{2C} \right\rceil$, which equals $\frac{\Delta+1}{2}$ as $\Delta < C$.

(2) $\Delta \geq C$ and $C \geq 3$ is odd (the case $C = 1$ is trivial by Lemma 2.3). For each edge $\{u, v\} \in M$, build a tree with C edges consisting of $\{u, v\}$, $\frac{C-1}{2}$ edges from E_u^+, and $\frac{C-1}{2}$ edges from E_v^+. Partition the remaining $\frac{\Delta-1}{2} - \frac{C-1}{2} = \frac{\Delta-C}{2}$ edges assigned to each vertex into $\left\lceil \frac{\Delta-C}{2C} \right\rceil$ stars with at most C edges. The number of occurrences of each vertex is

$$1 + \left\lceil \frac{\Delta - C}{2C} \right\rceil + \Delta - \frac{\Delta + 1}{2} = \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil.$$

(3) $\Delta \geq C$ and $C \geq 4$ is even (the case $C = 2$ is solved by Corollary 2.2). Build a tree with $C - 1$ edges consisting of $\{u, v\}$, $\frac{C-2}{2}$ edges from E_u^+, and $\frac{C-2}{2}$ edges from E_v^+. Partition the remaining $\frac{\Delta-1}{2} - \frac{C-2}{2} = \frac{\Delta-C+1}{2}$ edges assigned to each vertex into stars with at most C edges. The number of occurrences of each vertex is

$$1 + \left\lceil \frac{\Delta - C + 1}{2C} \right\rceil + \frac{\Delta - 1}{2} = \left\lceil \frac{\Delta(C+1) + 1}{2C} \right\rceil = \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil,$$

where the last equality holds because both Δ and $(C + 1)$ are odd. □

2.4.6 Towards a proof for the remaining cases

In this section, we describe an attempt to prove that the lower bound $\left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$ of Proposition 2.1 is attained in the remaining cases where $\Delta \geq 5$ is odd. We attempt to resolve the remaining cases by using induction and Tutte's matching theorem. We may assume that Δ is odd by Theorem 2.1.

The idea is to use the construction from Proposition 2.4 of Section 2.4.5, which solves the case where the graph contains a perfect matching, as a base case for a proof by induction. Then, if the graph does not contain a perfect matching, an easy consequence of Tutte's matching theorem shows that it contains an edge-cut of size at most $\Delta - 1$. We would then like to recurse on each side of the cut as we did in the proof of Theorem 2.5 and combine the edge-partitions of each side into a partition of the whole graph. However, as opposed to Theorem 2.5, it is more difficult to deal with the edges across the cut in this case.

We may orient each edge $e = \{u, v\}$ across the cut from u to v and let the side containing u decide which partition will contain e. To guarantee that v is not incident to too many subgraphs at the end, we can simply force v to be incident to one fewer subgraph in the edge-partition of the side containing v.

We note that, since the cuts have size less than Δ, it is possible to recursively orient the edges of cuts so that no side has more than $\Delta - 1$ edges pointing towards it (including edges from previous steps of the recursion).

However, it seems difficult to control the distribution of the edges pointing towards a side. If, for example, a single vertex v had $\Delta - 1$ edges pointing towards it, then it is clearly impossible to obtain the desired edge-partition, as v would need to be in a negative number of parts. On the other hand, if it were possible to control the distribution of the edges pointing towards a side, the following strengthening of Proposition 2.4 would be sufficient to prove the base case of the induction.

Definition 2.2 *G is* near-Δ-regular *if the vertices of G have degrees between $\frac{\Delta}{2}$ and Δ and $|E(G)| \geq \frac{\Delta}{2}(|V(G)| - 1) - 1$ (i.e., the total degree is off by at most $\Delta - 1$).*

Lemma 2.4 *Let $LB(C, \Delta) = \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$, the lower bound of Proposition 2.1. Let C, Δ be positive integers with Δ odd and $(\Delta - 1)/2$ not a multiple of C. Let G be a near-Δ-regular graph with girth at least 5 and a perfect matching. Then G has an edge-partition where each vertex v is incident to at most $LB(C, \Delta) - (\Delta - \mathbf{deg}(v))$ subgraphs of the partition.*

We note that it may be possible to first recursively find all the cuts and then orient the edges so that no vertex has more than $(\Delta - 1)/2$ edges pointing towards it. We also note that the above lemma is not in its strongest form (e.g., we could have the total degree differ by more the one stated in the lemma) but we clearly cannot relax the condition that every vertex v has degree greater than $\Delta/2$ (otherwise, v is contained in too many subgraphs even if we use stars of size C centered at v). We now prove Lemma 2.4.
Proof: Let M be a perfect matching in G. Since at most Δ vertices of G have degree not equal to Δ, at most Δ edges in M connect an odd degree vertex to an even degree vertex. Let G' be the graph obtained from G by removing edges of M matching odd degree vertices of G and adding (at most $\Delta/2$) edges to pair up the remaining odd degree vertices of G. Thus, G' is an even graph and we may obtain an Eulerian orientation O' of G'.

O' induces an orientation of some of the edges of G. We orient the remaining edges of G "both ways" and count half towards the in-degree and half towards the out-degree of the vertex. Let S be the set of vertices of G with degree less than Δ. We reverse some of the arcs of O so that all vertices in S have out-degree at least $\Delta/2$. This can be done greedily since G has no C_4 and we are only off by $\Delta - 1$ from the total degree. We call this new orientation O.

Let S' be the set of vertices with out-degree less than $\Delta/2$ in O. Note that the vertices of S' have degree Δ and out-degree at least $(\Delta - 3)/2$. Let $N^-(v)$ denote the set of vertices with an arc to v in O. Note that for two distinct vertices u and v in G, their neighborhood intersects in at most one vertex (since G has girth at least 5). Therefore $N^-(u) \cap N^-(v)$ also contains at most one vertex.

Therefore, we may find a subgraph H of the graph induced by the edges $N^-(v)$ to v for all v in S' with the following properties:

- Each vertex in $N^-(S')$ has degree at most 1.
- Each vertex in $s \in S'$ has degree at least $\delta^-(s) - (\Delta - 1)/2 \geq \delta^-(s) - (2C - 1)$, where $\delta^-(s)$ is the in-degree of s.

We say that a star or double-star is *full* if it contains exactly C edges.

Now, we can find a set $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$ of edge disjoint subgraphs of G such that

- $\mathcal{S}_1$ is a set of full double stars centered at the endpoints of unoriented edges in O,
- $\mathcal{S}_2$ is a set of full stars,
- $\mathcal{S}_3 = \{S_v\}_{v \in V(G)}$, where S_v is a star centered at v of size at most $C - 1$, and
- only stars in $\mathcal{S}_3$ contain edges of H.

These edge disjoint subgraphs can be found greedily by first finding $\mathcal{S}_1$ and then partitioning the out-edges of each vertex into sets of size C and a remainder set of edges of size $\leq C$.

Now, for each $s \in S'$, remove all but two stars centered at s in $\mathcal{S}_2$ and remove one star centered at s in $\mathcal{S}_3$. Let R be the set of edges removed in this way. For each edge

$e = \{u, v\} \in E(H)$, add an out-edge of v in R if there is any left (and remove this edge from R) to the star containing e. By the properties of H, no edges of R are left in the end.

We claim that this new set of subgraphs form a C-edge-partition where each vertex v is incident to at most $LB(C, \Delta) - (\Delta - \mathbf{deg}(v))$ partitions.

Indeed, the elements of $\mathcal{S}'$ are edge disjoint and have size at most C (since every star in $\mathcal{S}_3$ has size at most $C - 1$). The vertices $v \in V - S''$ are incident to

$$\begin{aligned}
\frac{\Delta+1}{2C} + \mathbf{deg}(v) - \frac{\Delta+1}{2} &= \frac{\Delta+1}{2C} + \frac{\Delta-1}{2} - (\Delta - \mathbf{deg}(v)) \\
&= \frac{\Delta(C+1)+1-C}{2C} - (\Delta - \mathbf{deg}(v)) \\
&\leq LB(C, \Delta) - (\Delta - \mathbf{deg}(v))
\end{aligned}$$

subgraphs if v is not incident to an unoriented edge, and

$$\begin{aligned}
1 + \left\lceil \frac{\Delta - C}{2C} \right\rceil + \mathbf{deg}(v) - \frac{\Delta+1}{2} &= 1 + \left\lceil \frac{\Delta - C}{2C} \right\rceil + \frac{\Delta-1}{2} - (\Delta - \mathbf{deg}(v)) \\
&= \left\lceil \frac{\Delta + C + C(\Delta-1)}{2C} \right\rceil - (\Delta - \mathbf{deg}(v)) \\
&= \left\lceil \frac{(C+1)\Delta}{2C} \right\rceil - (\Delta - \mathbf{deg}(v)) \\
&= LB(C, \Delta) - (\Delta - \mathbf{deg}(v))
\end{aligned}$$

subgraphs if v is incident to an unoriented edge. This satisfies the conditions in the lemma.

Recall that vertices in S' have out-degree at least $(\Delta - 3)/(2C)$. If $v \in S'$ and v is incident to an unoriented edge, v appears in

$$\begin{aligned}
\frac{\Delta - 3 - (2C-2)}{2C} + \mathbf{deg}(v) - \frac{\Delta-3}{2} &= \frac{\Delta - 1 - 2C}{2C} + \mathbf{deg}(v) - \frac{\Delta+1}{2} + \frac{4}{2} \\
&= \frac{\Delta-1}{2C} - 2 + \mathbf{deg}(v) - \frac{\Delta+1}{2} + 2 \\
&\leq \frac{\Delta+1}{2C} + \mathbf{deg}(v) - \frac{\Delta+1}{2} \\
&\leq LB(C, \Delta) - (\Delta - \mathbf{deg}(v))
\end{aligned}$$

subgraphs. If $v \in S'$ and and v is not incident to an unoriented edge, v appears in

$$\begin{aligned}
1 + \left\lceil \frac{\Delta - 3 - C - (2C-2)}{2C} \right\rceil + \mathbf{deg}(v) - \frac{\Delta-3}{2} &= 1 + \left\lceil \frac{\Delta - 1 - C}{2C} \right\rceil - 2 + \mathbf{deg}(v) - \frac{\Delta+1}{2} + 2 \\
&\leq 1 + \left\lceil \frac{\Delta - C}{2C} \right\rceil + \mathbf{deg}(v) - \frac{\Delta+1}{2} \\
&= LB(C, \Delta) - (\Delta - \mathbf{deg}(v))
\end{aligned}$$

subgraphs. Again, the conditions in the lemma are satisfied as required.

□

2.5 Conclusions

In this chapter we introduced the traffic grooming problem in unidirectional WDM rings when the request graph belongs to the class of graphs with maximum degree Δ. Such a model allows the network to support dynamic traffic without reconfiguring the electronic equipment. We showed that this problem is essentially equivalent to finding the least integer $M(C, \Delta)$ such that the edges of any graph with maximum degree at most Δ can be partitioned into subgraphs with at most C edges and each vertex appears in at most $M(C, \Delta)$ subgraphs. We established the value of $M(C, \Delta)$ for many cases, leaving open only the case where $\Delta \geq 5$ is odd, $\Delta \pmod{2C}$ is between 3 and $C - 1$, $C \geq 4$, and the graph does not contain a perfect matching. Table 2.1 summarizes what is known about $M(C, \Delta)$, including the case where the graph has a perfect matching. For the remaining cases, we hope to either extend the counterexample given in Section 2.4.2 or to complete the partial proof given in Section 2.4.6, which can be seen as a strengthening of Proposition 2.4.

Considering bounded-degree request graphs is natural from a networking perspective. It would be also interesting to consider as input other families of request graphs that make sense from a telecommunications point of view, like circulant graphs or graphs of bounded diameter.

$C \vert \Delta$	1	2	3	4	5	6	7	8	9	...	Δ even	Δ odd
1	1	2	3	4	5	6	7	8	9	...	Δ	Δ
2	1	2	3	3	4	5	6	6	7	...	$\lceil \frac{3\Delta}{4} \rceil$	$\lceil \frac{3\Delta}{4} \rceil$
3	1	2	3 (2)	3	4	5 (4)	5	6	7 (6)	...	$\lceil \frac{2\Delta}{3} \rceil$	$\lceil \frac{2\Delta+1}{3} \rceil$ ($\lceil \frac{2\Delta}{3} \rceil$)
4	1	2	2	3	4	4	5	5	6	...	$\lceil \frac{5\Delta}{8} \rceil$	$\geq \lceil \mathbf{\frac{5\Delta}{8}} \rceil$ (=)
5	1	2	2	3	4 (3)	4	5	5	6	...	$\lceil \frac{3\Delta}{5} \rceil$	$\geq \lceil \mathbf{\frac{3\Delta}{5}} \rceil$ (=)
6	1	2	2	3	$\geq$ **3** (=)	4	5	5	6	...	$\lceil \frac{7\Delta}{12} \rceil$	$\geq \lceil \mathbf{\frac{7\Delta}{12}} \rceil$ (=)
7	1	2	2	3	$\geq$ **3** (=)	4	5 (4)	5	6	...	$\lceil \frac{4\Delta}{7} \rceil$	$\geq \lceil \mathbf{\frac{4\Delta}{7}} \rceil$ (=)
8	1	2	2	3	$\geq$ **3** (=)	4	$\geq$ **4** (=)	5	6	...	$\lceil \frac{9\Delta}{16} \rceil$	$\geq \lceil \mathbf{\frac{9\Delta}{16}} \rceil$ (=)
9	1	2	2	3	$\geq$ **3** (=)	4	$\geq$ **4** (=)	5	6 (5)	...	$\lceil \frac{5\Delta}{9} \rceil$	$\geq \lceil \mathbf{\frac{5\Delta}{9}} \rceil$ (=)
...	...	...	...	...	...	...	...	...	...	...	...	...
C	1	2	2	3	$\geq$ **3** (=)	4	$\geq$ **4** (=)	5	$\geq$ **5** (=)	...	$\lceil \frac{C+1}{C} \frac{\Delta}{2} \rceil$	$\geq \lceil \mathbf{\frac{C+1}{C}} \frac{\Delta}{2} \rceil$ (=)

Table 2.1: Known values of $M(C, \Delta)$. The **bold** cases remain open. The cases in brackets only hold if the graph has a perfect matching. The symbol "(=)" means that the corresponding lower bound is attained.

Chapter 3

Bidirectional WDM Rings

In this chapter we study the minimization of ADMs (Add-Drop Multiplexers) in optical WDM bidirectional rings considering symmetric shortest path routing and all-to-all unitary requests. We precisely formulate the problem in terms of graph decompositions, and state a general lower bound for all the values of the grooming factor C and n, the size of the ring. We first study exhaustively the cases $C = 1$, $C = 2$, and $C = 3$, providing improved lower bounds, optimal constructions for several infinite families, as well as asymptotically optimal constructions and approximations. We then study the case $C > 3$, focusing specifically on the case $C = k(k+1)/2$ for some $k \geq 1$. We give optimal decompositions for several congruence classes of n, using the existence of a certain combinatorial design. We conclude with a comparison of the switching cost in unidirectional and bidirectional WDM rings.

Keywords: traffic grooming, SONET ADM, optical WDM network, graph decomposition, combinatorial designs.

3.1 Introduction

Whereas traffic grooming in unidirectional rings has been widely studied in the literature (see page 36), the bidirectional ring has not been studied from a graph partitioning approach. Filling this gap in the literature is the main objective of this chapter.

Namely, we focus on bidirectional rings with symmetric shortest path routing, and on the all-to-all case. We begin by formally stating the problem in terms of graph partitioning in Section 3.1.1. In Section 3.2 we provide lower bounds that improve those existing in the literature. The remainder of the chapter is devoted to find families of solutions for certain values of C and n. Namely, in Section 3.3 we show that the case $C = 1$ is relatively easy to solve. In Section 3.4 we study the case $C = 2$, improving the general lower bound and providing a $\frac{34}{33}$-approximation for all values of n. In Section 3.5 we tackle the case $C = 3$, improving the general lower bound and giving families of optimal and asymptotically

optimal solutions for all values of n. In Section 3.6 we study the case $C > 3$, by either improving the general lower bound or providing families of optimal solutions when C is of the form $1+2+\ldots+k$. In Section 3.7 we compare the lower bounds for the switching cost in unidirectional and bidirectional rings. We conclude the chapter in Section 3.8. We first state precisely the problem we study.

3.1.1 Statement of the problem

Load constraint. In a graph-theoretical approach, we are given an optical network represented by a directed graph G on n vertices (in many cases a symmetric one) – called the *physical graph* –, for example a unidirectional ring $\vec{C}_n$ or a bidirectional symmetric ring C_n^*. We are given also a traffic (or instance) matrix, that is a family of connection requests represented by an arc-weighted multidigraph I – called the *logical* or *request graph* – where the number of arcs from i to j corresponds to the number of requests from i to j, and the weight of each arc corresponds to the amount of bandwidth used by each request. Here we suppose that there is exactly one request from i to j (all-to-all case) and that each request uses the same bandwidth. In that case $I = K_n^*$. We also suppose that the bandwidth used by any request is a fraction $1/C$ of the available bandwidth of a wavelength. Said otherwise, each wavelength λ can carry on a given arc at most C requests. This positive integer C is called the *grooming factor*. For a wavelength λ, we denote by B_λ the set of requests carried by λ. Satisfying a request r from i to j consists in finding a dipath $P(r)$ in G and assigning it a wavelength λ. Note that a wavelength λ is directed either clockwise or counterclockwise, so all the dipaths associated to requests in a same B_λ are directed in the same way.

For a subgraph B_λ of requests of I, we define the *load* of an arc e of G, $L(B_\lambda, e)$, as the number of requests which are routed through e, that is

$$L(B_\lambda, e) := |\{P(r); r \in E(B_\lambda); e \in P(r)\}|.$$

Note that if B_λ is associated to a clockwise (resp. counterclockwise) wavelength λ, only the clockwise (resp. counterclockwise) arcs of the ring are loaded by B_λ. The constraint given by the grooming factor C means that for each subgraph B_λ and each arc e, $L(B_\lambda, e)$ is at most C. In this chapter we focus on the bidirectional ring topology with all-to-all unitary requests. Therefore, our problem consists in finding a partition of K_n^* into subdigraphs B_λ satisfying the load constraint for C_n^* and such that the total number of vertices is minimized. We have two choices for routing a request (i, j): either clockwise or counterclockwise. Although there is no physical constraint imposing it, it is common for the operator to consider symmetric routings. That is, if the request (i, j) is routed clockwise, then the request (j, i) is routed counterclockwise. Furthermore it is also common for the sake of simplicity to use shortest path routing. Therefore we will restrict ourselves to symmetric shortest path routings. Let us see how the restrictions on the routing affect the solutions.

Constraints on the routing. In a ring C_n^* with an odd number of vertices, shortest path routing implies symmetric routing. But in a ring with an even number of vertices

this is not necessarily the case, as a request of the form $(i, i + \frac{n}{2})$ can be routed via a shortest path in both directions. Consider for example $n = 4$ and $C = 2$. If we do not impose symmetric routing, we can have a solution consisting of the two subdigraphs B_{λ_1} with the requests $(0,1)$, $(1,2)$, $(2,3)$, $(3,0)$, $(0,2)$, and $(2,0)$ routed clockwise, and B_{λ_2} with the requests $(1,0)$, $(0,3)$, $(3,2)$, $(2,1)$, $(1,3)$, and $(3,1)$ routed counterclockwise. Altogether we use 8 ADMs. Suppose now that we further impose symmetric routing, and assume without loss of generality that the requests $(0,2)$ and $(1,3)$ are routed clockwise. The best we can do for a B_λ with 4 vertices is to put 5 requests if λ is clockwise, namely $(0,1)$, $(1,2)$, $(2,3)$, $(3,0)$, and at most one of $(0,2)$ and $(1,3)$. The other request out of $(0,2)$ and $(1,3)$ will need 2 ADMs, so we use a total of 12 ADMs. If we do not use any B_λ with 4 vertices, note that a subdigraph with 3 (resp. 2) vertices contains at most 3 requests (resp. 1 request). Therefore to route all the requests we need at least 12 ADMs.

Imposing shortest path routing might increase the number of ADMs of an optimal solution. Consider for example $n = 3$ and $C = 3$. With shortest path routing, we need two subdigraphs B_{λ_1} with the requests $(0,1)$, $(1,2)$, $(2,0)$ and B_{λ_2} with the requests $(1,0)$, $(2,1)$, $(0,2)$, for a total of 6 ADMs (each arc of C_3^* is loaded once). Without the constraint of shortest path routing, we can do it with 3 ADMs, namely with all the requests routed clockwise. In that case, the requests $(1,0)$, $(2,1)$, and $(0,2)$ are routed via dipaths of length 2 (for instance, the request $(1,0)$ used the arcs $(1,2)$ and $(2,0)$). In that case the load of the arcs (in the clockwise direction) is 3.

We cannot always use shortest path routing and have a minimum load. Indeed, consider the case $C = 1$ and a set of 3 requests (i,j), (j,k), and (k,i) forming a triangle. The subdigraph formed by the 3 requests routed in the same direction has load 1, but there is not reason that the associated routes are shortest paths. For example, let $n = 5$ and $(0,1)$, $(1,2)$, $(2,0)$ be the three mentioned requests, which we assume to be routed clockwise. If we want a valid solution, then the request $(2,0)$ is routed via the path $[2,3,4,0]$ of length 3 (and not 2). If we want to use shortest paths, then these three requests induce load 2, hence they cannot fit together in the same wavelength. Summarizing, in this example either we use shortest paths and the load is 2 or we get a solution with load one but not using shortest paths.

Symmetric shortest path routing. However, in the sequel of the chapter we will only consider symmetric shortest path routing. Besides being a common scenario in telecommunication networks, this assumption also simplifies the problem, as we can split it into two separate problems, half of the requests being routed clockwise and half counterclockwise. Each of these two subproblems can be viewed as a grooming problem where $G = \vec{C}_n$ (the unidirectional cycle) and $I = T_n$, where T_n is a tournament on n vertices, that is, a complete oriented graph (for each pair of vertices $\{i,j\}$ there is exactly one of the arcs (i,j) or (j,i)).

As we consider shortest path routing, for n odd T_n is unique. But for n even we have two possibilities for the pairs of the form $\{i, i + \frac{n}{2}\}$: either the arc $(i, i + \frac{n}{2})$ or $(i + \frac{n}{2}, i)$. So the choice of these arcs has to be made. We are ready to state precisely our problem.

Traffic Grooming in Bidirectional Rings
Input: A unidirectional cycle $\vec{C}_n$ with vertices $0, \ldots, n-1$, a grooming factor C and a digraph of requests consisting of the tournament T_n with arcs $(i, i+1)$ for $0 \le i \le n-1$ and $1 \le q \le \frac{n-1}{2}$, plus if n is even $\frac{n}{2}$ arcs of the form $(i, i+\frac{n}{2})$, where we cannot have both $(i, i+\frac{n}{2})$ and $(i+\frac{n}{2}, i)$ (or said otherwise, for n even we have one of the two arcs $(i, i+\frac{n}{2})$ or $(i+\frac{n}{2}, i)$ for $0 \le i \le \frac{n}{2}-1$).
Output: A partition of T_n into digraphs B_λ, $1 \le \lambda \le \Lambda$, such that for each arc $e \in E(\vec{C}_n)$, $L(B_\lambda, e) \le C$.
Objective: Minimize $\sum_{\lambda=1}^{\Lambda} |V(B_\lambda)|$. The minimum will be denoted $A(C, n)$.

Remark 3.1 *Solutions to the original problem can be found by solving the above problem and using the solution for the counterclockwise requests by reversing the orientation of the arcs of $\vec{C}_n$ and T_n. Therefore, the total number of ADMs for the original problem – under the constraints of symmetric shortest path routing – is $2A(C, n)$.*

Let us see an example for $n = 5$ and $C = 1$. Then the following three subdigraphs form a solution with 10 ADMs: one with arcs $(0,1),(1,3),(3,0)$, another with arcs $(1,2),(2,4),(4,1)$, and another with arcs $(0,2),(2,3),(3,4),(4,0)$. A solution for the bidirectional ring C_5^* and $I = K_n^*$ uses 20 ADMs.

Let now $n = 5$ and $C = 2$. We can use the preceding solution or another one with also 10 ADMs with only two $\vec{C}_5$'s with arcs $(0,2),(1,2),(2,3),(3,4),(4,5)$ and $(0,2),(2,4),(4,1),(1,3),\ (3,0)$, the second one inducing load 2. But we can do better, with only 8 ADMs, with one subdigraph with arcs $(1,3),(3,4),(4,1)$, and another one with arcs $(0,1),(1,2),(0,2),(2,3),\ (2,4),(3,0),(4,0)$. This latter partition is optimal.

To tackle our problem we will use tools from design theory, similar to those used for the unidirectional ring and $I = K_n$ [48, 49]. In particular, it is helpful to use for a given C digraphs having a maximum ratio number of arcs over number of vertices (see Section 3.2.2).

Admissible digraphs. Let $B_\lambda = (V_\lambda, E_\lambda)$ be a digraph with $V_\lambda = \{a_0, \ldots, a_{p-1}\}$ involved in a partition of the tournament T_n. Note that the edges of B_λ belong to T_n, so $(a_i, a_j) \in E_\lambda$ if and only $d_{\vec{C}_n}(a_i, a_j) \le \frac{n}{2}$, where $d_{\vec{C}_n}(a_i, a_j)$ is the distance between a_i and a_j in $\vec{C}_n$.

A digraph B_λ is said to be *admissible* if it satisfies the load constraint, that is, $L(B_\lambda, e) \le C$ for each arc $e \in E(\vec{C}_n)$. A partition of T_n into admissible subdigraphs is called *valid*. As the paths associated to an arc of B_λ form a dipath (an interval) in $\vec{C}_n$, the load is exactly the same as if we consider B_λ embedded in a cycle $\vec{C}_p$ with vertex set $0, 1, \ldots, p-1$. More precisely, we associate to B_λ the digraph B_λ^p with vertices $0, 1, \ldots, p-1$ and with $(i, j) \in E(B_\lambda^p)$ if and only if $(a_i, a_j) \in E(B_\lambda)$. Hence, to compute the load we will consider digraphs with p vertices and their load in the associated $\vec{C}_p$. Note that it can happen that $d_{\vec{C}_n}(a_i, a_j) \le \frac{n}{2}$ but $d_{\vec{C}_p}(i, j) > \frac{p}{2}$, and viceversa.

Figure 3.1(a) illustrates a digraph B_λ that is admissible for $n = 8$ and $C = 2$, as it induces load 2 in $\vec{C}_8$. Its associated digraph B_λ^4 is shown in Figure 3.1(b). Figure 3.1(c) shows a digraph B'_λ which has also B_λ as associated digraph, but it is not admissible as (a_3, a_0) is not an arc of T_8.

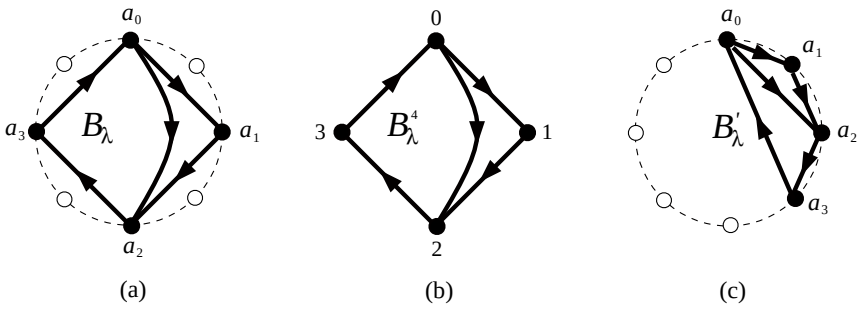

Figure 3.1: (a) Digraph B_λ admissible for $n = 8$ and $C = 2$; (b) Its associated digraph B^4_λ; (c) Non-admissible digraph B'_λ that has also B^4_λ as associated digraph.

Figure 3.2(a) shows and admissible digraph for $n = 7$ and $C = 2$. Its associated digraph B^5_λ, which is depicted in Figure 3.2(b), induces load 2 but the arc $(1, 4)$ is not routed via a shortest path (although the arc (a_1, a_4) was in B_λ).

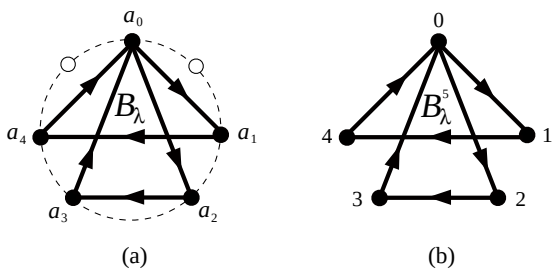

Figure 3.2: (a) Digraph B_λ admissible for $n = 7$ and $C = 2$; (b) Its associated digraph B^5_λ.

In what follows we will compute the load in the associated digraph, but we will have to be careful that the arcs of B_λ are those of T_n, as pointed out by the above examples.

3.2 Lower Bounds

In this section we state general lower bounds on the number of ADMs used by any solution.

3.2.1 Equations of the problem

Given a valid solution of the problem, let a_p denote the number of subgraphs of the partition with exactly p nodes, let A denote the total number of ADMs, let Λ denote the number of subgraphs of the partition, and let E_λ be the set of arcs of B_λ. Recall that here

$I = T_n$, which has $\frac{n(n-1)}{2}$ arcs. The following equalities hold:

$$A = \sum_{p=2}^{n} p a_p \tag{3.1}$$

$$\sum_{p=2}^{n} a_p = \Lambda \tag{3.2}$$

$$\sum_{\lambda=1}^{\Lambda} |E_\lambda| = \frac{n(n-1)}{2} \tag{3.3}$$

Proposition 3.1 *For $I = T_n$,*

$$\Lambda \geq \left\lceil \frac{n^2 + \alpha}{8C} \right\rceil, \text{ where } \alpha = \begin{cases} -1, & \text{if } n \text{ is odd} \\ 4, & \text{if } n \equiv 2 \pmod 4 \\ 8, & \text{if } n \equiv 0 \pmod 4 \end{cases}$$

Proof: The set of arcs of T_n of the form $(i, i+q)$, $0 \leq q < \frac{n}{2}$, load each arc of the ring exactly q times. So if n is odd the load of any arc of the ring is $1 + 2 + \ldots + \frac{n-1}{2} = \frac{n^2-1}{8}$.

If n is even the load due to these arcs is $1 + 2 + \ldots + \frac{n-2}{2} = \frac{n^2-2n}{8}$. We have to add the load due to arcs of T_n of the form $\left(i, i + \frac{n}{2}\right)$. As there are $\frac{n}{2}$ such arcs, the total load is $\frac{n^2}{4}$ and so one arc of the ring has load at least $\frac{n}{4}$.

If $n \equiv 2 \pmod 4$ that gives a load at least $\left\lceil \frac{n}{4} \right\rceil = \frac{n+2}{4}$, so one arc has load at least $\frac{n^2-2n}{8} + \frac{n+2}{4} = \frac{n^2+4}{8}$.

If $n \equiv 0 \pmod 4$ the maximum load due to the arcs $\left(i, i + \frac{n}{2}\right)$ is at least $\frac{n}{4}$, but in this case we can give a better bound. Indeed, suppose w.l.o.g. that we have the arc $\left(0, \frac{n}{2}\right)$, and let j be the number of arcs starting in the interval $[1, \frac{n}{2} - 1]$ of the form $\left(i, i + \frac{n}{2}\right)$ with $0 < i < \frac{n}{2}$. The load of the arc $\left(\frac{n}{2} - 1, \frac{n}{2}\right)$ of the ring is then $j + 1$. As there are $\frac{n}{2} - 1 - j$ arcs ending in the interval $[1, \frac{n}{2} - 1]$, the load of the arc $(0, 1)$ is $1 + \frac{n}{2} - 1 - j$. Therefore the sum of the loads of the arcs $(0, 1)$ and $\left(\frac{n}{2} - 1, \frac{n}{2}\right)$ is $\frac{n}{2} + 1$, and so one of these 2 arcs has load $\left\lceil \frac{n}{4} + \frac{1}{2} \right\rceil = \frac{n}{4} + 1$. The total load of this arc is $\frac{n^2-2n}{8} + \frac{n}{4} + 1 = \frac{n^2+8}{8}$.

As each subgraph can load one arc at most C times, we obtain the lemma. □

3.2.2 The parameter $\gamma(C, p)$

To obtain accurate lower bounds we need to bound the value of $|E_\lambda|$ for a digraph with $|V_\lambda| = p$ vertices, satisfying the load constraint (admissible digraph). As we discussed in the preceding section, we need only to consider the associated digraph embedded in $\vec{C}_p$. To this end, we introduce the following definition.

Definition 3.1 *Let $\gamma(C, p)$ be the maximum number of arcs of a digraph H with p vertices such that $L(H, e) \leq C$, for every arc e of $\vec{C}_p$.*

The next lemma gives the value of $\gamma(C,p)$ and shows that, in fact, the maximum number of requests we can groom is attained by taking those of minimum length. It is worth to mention that this property is not true if the physical graph is a path, as shown with a counterexample in [44].

Proposition 3.2 *Let $C = \frac{k(k+1)}{2} + r$, with $0 \le r \le k$. Then*

$$\gamma(C,p) = \begin{cases} \frac{p(p-1)}{2} & , \text{ if } p \le 2k+1, \text{ or } p = 2k+2 \text{ and } r \ge \frac{k+2}{2} \\ kp + 2r - 1 & , \text{ if } p = 2k+2 \text{ and } 1 \le r < \frac{k+2}{2} \\ kp + \left\lfloor \frac{rp}{k+1} \right\rfloor & , \text{ otherwise} \end{cases}$$

The graphs achieving $\gamma(C,p)$ are either the tournament T_p if p is small (namely, if $p \le 2k+1$ or $p = 2k+2$ and $r \ge \frac{k+2}{2}$), or subgraphs of a circulant digraph containing all the arcs of length $1,2,\ldots,k$, plus some arcs of length $k+1$ if $r > 0$.

Proof: We distinguish three cases according to the value of p.

Case 1. If p is small, that is such that the tournament T_p loads each arc at most C times, then $\gamma(C,p) = \frac{p(p-1)}{2}$. Let us now see for which values of p this fact holds.

If p is odd, the load of T_p is $\frac{p^2-1}{8} \le C$. The inequality $p^2-1 \le 8C$ implies $p^2-1 \le 4k(k+1)+8r$, and is satisfied if $p \le 2k+1$, as $p^2 - 1 \le 4k(k+1)$.

If p is even, the load of T_p is $\frac{p^2}{8} + \frac{1+\delta}{2}$, where $\delta = 1$ if $p \equiv 0 \pmod 4$ (see proof of Proposition 3.1).

If $p \le 2k$, it holds $\frac{p^2+8}{8} \le \frac{4k^2+8}{8} \le \frac{k(k+1)}{2} \le C$.

For $p = 2k+2$, it holds $\frac{p^2}{8} + \frac{1+\delta}{2} = \frac{k^2}{2} + k + 1 + \frac{\delta}{2} \le \frac{k^2+k}{2} + r = C$ if and only if $r \ge \frac{k+2+\delta}{2}$, with $\delta = 1$ if $p \equiv 0 \pmod 4$, that is, if k is odd. Therefore, the condition is satisfied if $r \ge \frac{k+2}{2}$.

In the next two cases, we provide first a lower bound on $\gamma(C,p)$, and then we prove a matching upper bound.

Case 2. If $p = 2k+2$ and $1 \le r < \frac{k+2}{2}$, a solution is obtained by taking all the arcs of length $1,2,\ldots,k\left(= \frac{p-2}{2}\right)$ – giving a load of $\frac{k(k+1)}{2}$ – plus $2r-1$ arcs of length $\frac{p}{2}$. For example, we can take the arcs $\left(i, i+\frac{p}{2}\right)$ for $i = 0,2,\ldots,2r-2\left(< \frac{p}{2}\right)$ and the arcs $\left(i, i-\frac{p}{2}\right)$ for $i = 1,3,\ldots,2r-3$. The load due to these arcs is at most r. Therefore, in this case $\gamma(C,p) \ge kp + 2r - 1$.

Case 3. If $p > 2k+2$ or $p = 2k+2$ and $r = 0$, a solution is obtained by taking all the arcs of length $1,2,\ldots,k$ plus $\left\lfloor \frac{rp}{k+1} \right\rfloor$ arcs of length $k+1$, in such a way that the load due to these arcs is at most C, which is always possible (for example, if p is prime with $k+1$, we take the requests $((k+1)i,(k+1)(i+1))$ for $0 \le i \le \left\lfloor \frac{rp}{k+1} \right\rfloor - 1$, the indices being taken modulo p). Therefore, in this case

$$\gamma(C,p) \ge kp + \left\lfloor \frac{rp}{k+1} \right\rfloor. \tag{3.4}$$

Let us now turn to upper bounds. Suppose we have a solution with γ arcs, γ_i being of length i on $\vec{C}_p$. As each arc of length i loads i arcs, and the total load of the arcs of $\vec{C}_p$ is

at most Cp, we have that

$$\begin{aligned} Cp &\geq \sum_{i=1}^{\infty} i\gamma_i \geq \sum_{i=1}^{k} i\gamma_i + (k+1)\left(\gamma - \sum_{i=1}^{k}\gamma_i\right) \\ &= \sum_{i=1}^{k} ip + (k+1)(\gamma - kp) + \sum_{i=1}^{k} \underbrace{(k+1-i)(p-\gamma_i)}_{\geq 0} \\ &\geq \frac{k(k+1)}{2}\cdot p + (k+1)(\gamma - kp). \end{aligned}$$

Since $Cp = \frac{k(k+1)}{2}\cdot p + rp$, we obtain $rp \geq (k+1)(\gamma - kp)$, and therefore

$$\gamma(C,p) \leq kp + \frac{rp}{k+1}\ . \tag{3.5}$$

Combining Equations (3.4) and (3.5), we get the result for case 3. For case 2, i.e., when $p = 2k+2$ and $1 \leq r < \frac{k+2}{2}$, Equation (3.5) yields $\gamma(C,p) \leq kp + 2r$. If we have equality, then necessarily $\gamma_i = p$ for $i = 1,\ldots,k$, so we have all arcs of length at most k. However, the $2r$ arcs of length at least $k+1$ induce a load at least $r+1$ on some arc of $\vec{C}_p$, so the total load would be strictly greater than C. Therefore, we have at most $\gamma(C,p) \leq kp + 2r - 1$, which gives the result. □

We define the parameter $\rho(C) = \max_p \left\{\frac{\gamma(C,p)}{p}\right\}$.

Proposition 3.3 *Let $C = k(k+1)/2 + r$, with $0 \leq r \leq k$. Then $\rho(C) \leq k + r/k + 1$.*

Proof: In Case 1 of the proof of Proposition 3.2, $\rho(C) \leq \frac{p-1}{2}$. If $p \leq 2k+1$, $\rho(C) \leq k$. If $p = 2k+2$ and $r \geq \frac{k+2}{2}$, $\rho(C) = k + \frac{1}{2} < k + \frac{r}{k+1}$. Otherwise, by Equation (3.5),

$$\rho(C) \leq \frac{kp + \frac{rp}{k+1}}{p} = k + \frac{r}{k+1}, \tag{3.6}$$

where $C = \frac{k(k+1)}{2} + r$, with $0 \leq r \leq k$. So, in all cases, $\rho(C) \leq k + \frac{r}{k+1}$. □

Table 3.1 shows the parameter $\gamma(C,p)$ for small values of C and p, as well as the parameter $\rho(C)$.

3.2.3 General lower bounds

By Propositions 3.1 and 3.2, Equations (3.1), (3.2), and (3.3) become

$$A = \sum_{p=2}^{n} pa_p \tag{3.7}$$

$$\sum_{p=2}^{n} a_p \geq \left\lceil \frac{n^2+\alpha}{8C} \right\rceil, \text{ where } \alpha = \begin{cases} -1 & \text{, if } n \text{ is odd} \\ 4 & \text{, if } n \equiv 2 \pmod 4 \\ 8 & \text{, if } n \equiv 0 \pmod 4 \end{cases} \tag{3.8}$$

$$\sum_{p=2}^{n} a_p\gamma(C,p) \geq \frac{n(n-1)}{2} \tag{3.9}$$

p	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\rho(C)$
$C = 1$	1	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	1
$C = 2$	1	3	5	7	9	10	**12**	13	**15**	16	**18**	19	**21**	22	**24**	3/2
$C = 3$	1	3	6	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**	**30**	**32**	2
$C = 4$	1	3	6	10	13	16	18	**21**	23	25	**28**	30	32	**35**	37	7/3
$C = 5$	1	3	6	10	15	18	21	**24**	26	29	**32**	34	37	**40**	42	8/3
$C = 6$	1	3	6	10	15	**21**	**24**	**27**	**30**	**33**	**36**	**39**	**42**	**45**	**48**	3
$C = 7$	1	3	6	10	15	21	25	29	32	35	**39**	42	45	48	**52**	13/4
$C = 8$	1	3	6	10	15	21	27	31	35	38	**42**	45	49	52	**56**	14/4
$C = 9$	1	3	6	10	15	21	28	33	37	41	**45**	48	52	56	**60**	15/4
$C = 10$	1	3	6	10	15	21	28	**36**	**40**	**44**	**48**	**52**	**56**	**60**	**64**	4

Table 3.1: The parameter $\gamma(C, p)$ for some values of C and p, as well as $\rho(C)$. The **bold** values achieve $\rho(C)$.

We are ready to prove the general lower bound on the number of ADMs used by any solution.

Theorem 3.1 (General Lower Bound) *Let $C = \frac{k(k+1)}{2} + r$, with $0 \le r \le k$. The number of ADMs required in a bidirectional ring with n nodes and grooming factor C satisfies*

$$A(C,n) \;\ge\; \left\lceil \frac{n(n-1)}{2 \cdot \rho(C)} \right\rceil \;=\; \left\lceil \frac{n(n-1)}{2} \frac{k+1}{k(k+1)+r} \right\rceil. \tag{3.10}$$

Proof: Using Equations (3.7) and (3.9), and the definition of $\rho(C)$, we get that the number A of ADMs used by any solution satisfies

$$\frac{n(n-1)}{2} \;\le\; \sum_{p=2}^{n} a_p \cdot \gamma(C,p) \;=\; \sum_{p=2}^{n} p \cdot a_p \cdot \rho(C) \;=\; \rho(C) \cdot A.$$

From the above equation and using Equation (3.6), we get

$$A \;\ge\; \left\lceil \frac{n(n-1)}{2 \cdot \rho(C)} \right\rceil \;=\; \left\lceil \frac{n(n-1)}{2} \frac{k+1}{k(k+1)+r} \right\rceil.$$

□

To achieve the lower bound of Theorem 3.1, the only possibility is to use graphs on p vertices with $\gamma(C,p)$ arcs. The **bold** values in Table 3.1 achieve $\rho(C)$, and therefore the subgraphs corresponding to those values (which exist by Proposition 3.2) are good candidates to construct an optimal partition of the request graph.

Comparison of existing lower bounds. In [74] the Ring Traffic Grooming problem in the bidirectional ring is studied. The authors state a lower bound regardless of routing for a general set of requests. In the particular case of uniform traffic, they get a

lower bound of $\frac{n^2-1}{4\sqrt{2C}}$ (see [74, Theorem 1, page 198]). They indicate in their article that they can improve this bound by a factor of 2 for all-to-all uniform unitary traffic. We thank T. Chow and P. Lin for sending us the proof of the following theorem, which is only announced in [74]:

Theorem 3.2 ([73, 74]) *If a traffic instance of ring grooming is uniform and unitary, then, regardless of routing,*

$$A(C,n) \geq \frac{1}{2\sqrt{C}} \sqrt{\frac{n^2(n-1)^2}{2} - n(n-1)}.$$

The lower bound we obtained in Theorem 3.1 is greater than the bound of Theorem 3.2, but it should be observed that we restrict ourselves to shortest path symmetric routing. Our bound is $\frac{n(n-1)}{2\rho(C)}$ and the lower bound of Theorem 3.2 is less than $\frac{n(n-1)}{2\sqrt{2C}}$. The fact that our bound is better follows from the fact that $\rho(C) < \sqrt{2C}$. Indeed,

$$\rho^2(C) \leq \left(k + \frac{r}{k+1}\right)^2 = k^2 + \frac{2kr}{k+1} + \frac{r^2}{(k+1)^2} < k^2 + 2r + 1 < k^2 + k + 2r = 2C.$$

3.3 Case $C = 1$

For $C = 1$, by Proposition 3.2 $\gamma(1,p) = p$ if $p \geq 2$. Furthermore, all the directed cycles achieve $\rho(1)$ (see Table 3.1).

Theorem 3.3

$$A(1,n) = \begin{cases} \frac{n(n-1)}{2} & , \text{ if } n \text{ is odd} \\ \frac{n^2}{2} & , \text{ if } n \text{ is even} \end{cases}$$

Proof: For $C = 1$, the only possible subgraphs involved in the partition of the edges of T_n are cycles and paths. If only cycles are used, the total number of ADMs is $\frac{n(n-1)}{2}$, which equals the lower bound of Theorem 3.1. Each path involved in the partition adds one unity of cost with respect to $\frac{n(n-1)}{2}$.

If $n = 2q+1$ is odd, by [52, Theorem 3.3] we know that the arcs of T_n can be covered with q $\vec{C}_3$'s and $\frac{q(q-1)}{2}$ $\vec{C}_4$'s. The total number of vertices of this construction is $3q + 2q(q-1) = q(2q+1) = \frac{n(n-1)}{2}$.

If n is even, each vertex must appear with odd degree in at least one subgraph, so the number of paths in any construction is at least $n/2$. Therefore, the lower bound becomes $\frac{n(n-1)}{2} + \frac{n}{2} = \frac{n^2}{2}$. By [52, Theorem 3.4] the arcs of T_n can be covered with

- 4 $\vec{C}_3$'s and $2q^2 - 3$ $\vec{C}_4$'s, if $n = 4q$ with $q > 1$;
- 2 $\vec{C}_3$'s and $2q^2 + 2q - 1$ $\vec{C}_4$'s, if $n = 4q + 2$.

For $n = 4$, we cover T_4 with a $\vec{C}_4$ and two arcs. Note that in these constructions, some arcs are covered more than once. In both cases, the total number of vertices of the construction is $\frac{n^2}{2}$, hence the lower bound is attained.

Finally, one can check that in the constructions of [52], the length of the arcs involved in the covering of T_n is in all cases bounded above by $\left\lfloor \frac{n}{2} \right\rfloor$, and therefore all the cycles induce load 1. □

Remark 3.2 *For the original problem with $G = C_n^*$ and $I = K_n^*$, if we apply Theorem 3.3 we get in the case $n/2$ a value of n^2 ADMS; but if we delete the constraint of symmetric routings we get a value of $n(n-1)/2$ by using [52, Theorems 4.1 and 4.2] (however these constructions use many K_2's).*

3.4 Case $C = 2$

When $C = 2$ the general lower bound of Theorem 3.1 gives $A(2, n) \geq \frac{n(n-1)}{3}$. We first improve this bound in Section 3.4.1, and then give solutions with a good approximation ratio in Section 3.4.2.

3.4.1 Improved lower bounds

For $C = 2$, by Proposition 3.2 $\gamma(2, 2) = 1$, $\gamma(2, 3) = 3$, $\gamma(2, 4) = 5$ (note that $\gamma(2, 4) = 6$ if the routing is not restricted to be symmetric), and $\gamma(2, p) = \left\lfloor \frac{3p}{2} \right\rfloor$ for $p \geq 5$. The optimal solutions for $p \geq 4$ even consist of the p arcs of length 1 $(i, i+1)$ for $0 \leq i \leq p-1$, plus the $p/2$ arcs of length 2 $(2i, 2i+2)$ for $0 \leq i \leq p/2 - 1$ (in fact, triangles sharing a vertex; see Figure 3.3 for $p = 6$). For p odd we have two classes of optimal graphs (see Figure 3.3 for $p = 5$).

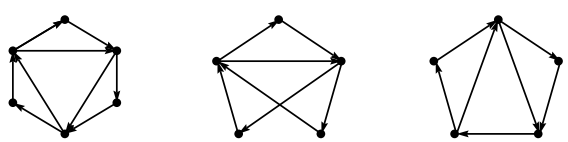

Figure 3.3: Some admissible digraphs for $C = 2$.

Equation (3.9) becomes in the case $C = 2$

$$\sum_{p=2}^{n} a_p \gamma(2, p) = a_2 + 3a_3 + 5a_4 + 7a_5 + 9a_6 + 10a_7 + 12a_8 + \ldots \geq \frac{n(n-1)}{2}.$$

Therefore,

$$A = \sum_{p=2}^{n} p a_p \geq \frac{2}{3} \sum_{p=2}^{n} a_p \gamma(2, p) + \frac{4}{3} a_2 + a_3 + \frac{2}{3} a_4 + \frac{1}{3}(a_5 + a_7 + a_9 + \ldots) \quad (3.11)$$

$$\geq \frac{n(n-1)}{3} + \frac{4}{3} a_2 + a_3 + \frac{2}{3} a_4 + \frac{1}{3}(a_5 + a_7 + a_9 + \ldots). \quad (3.12)$$

We can already see that the bound $\frac{n(n-1)}{3}$ cannot be attained. Indeed, to reach it we need to use only graphs with $6, 8, 10, \ldots$ vertices. But the number of graphs Λ satisfies, by Proposition 3.1, $\Lambda \geq \frac{n^2-1}{16}$, so $A \geq 6\frac{n^2-1}{16} > \frac{n(n-1)}{3}$.

The following proposition gives a lower bound of order $\frac{11}{32}n(n-1)$. Note that $11/32 > 11/33 = 1/3$.

Proposition 3.4 (Tighter Lower Bound for $C = 2$)

$$A(2,n) \geq \left\lceil \frac{11n^2 - 8n - 3}{32} \right\rceil = \left\lceil \frac{11}{16}\frac{n(n-1)}{2} + \frac{3n-3}{32} \right\rceil. \tag{3.13}$$

Proof: We can write $A \geq 6(\Lambda - a_2 - a_3 - a_4 - a_5) + 2a_2 + 3a_3 + 4a_4 + 5a_5$, that is,

$$A \geq 6\Lambda - (4a_2 + 3a_3 + 2a_4 + a_5). \tag{3.14}$$

From Equations (3.11) and (3.12) we get that

$$3A \geq n(n-1) + (4a_2 + 3a_3 + 2a_4 + a_5). \tag{3.15}$$

Summing Equations (3.14) and (3.15) gives

$$4A \geq 6\Lambda + n(n-1). \tag{3.16}$$

By Proposition 3.1, we have that

$$\Lambda \geq \frac{n(n-1)}{16} + \frac{n+\alpha}{16}. \tag{3.17}$$

Combining Equations (3.16) and (3.17) and using that $\alpha \geq -1$ yields

$$A \geq \frac{11n(n-1)}{32} + \frac{3n}{32} + \frac{3\alpha}{32} \geq \frac{11n^2 - 8n - 3}{32}.$$

□

3.4.2 Upper bounds

In this section we build families of solutions for $C = 2$. We conjecture that there exists a decomposition using A vertices with ratio $\frac{A}{\frac{n(n-1)}{2}}$ of order $\frac{11}{16}$, which would be optimal by Proposition 3.4. For that, we should find some (multipartite) graphs achieving this ratio. A candidate is $K_{4,4,4}$, which has 48 edges. Unfortunately, we have not been able to cover it with 33 vertices (which would achieve the optimal ratio) but only with 34, giving a 34/33-approximation.

For the sake of the presentation, we first present a simple 12/11-approximation inspired from a construction of [52].

A 12/11-approximation

This construction is defined recursively. Suppose we have a solution for n vertices using A_n ADMs, with $n = 2p$ or $n = 2p+1$. Let the vertex set be labeled $0_A < 1_A < \ldots < (p-1)_A < 0_B < 1_B < \ldots < (p-1)_B$, plus ∞ is n is odd. For $n+2$, we add two vertices x_A and x_B with the order $x_A < 0_A < 1_A < \ldots < (p-1)_A < x_B < 0_B < 1_B < \ldots < (p-1)_B < \infty$. We use as subdigraphs those of the solution for n plus the $\lfloor p/2 \rfloor$ digraphs on the 6 vertices $x_A, i_A, (i+\lfloor p/2 \rfloor)_A, x_B, i_B, (i+\lfloor p/2 \rfloor)_B$ and the 8 arcs (x_A, i_A), $(x_A, (i+\lfloor p/2 \rfloor)_A)$, (i_A, x_B), $((i+\lfloor p/2 \rfloor)_A, x_B)$, (x_B, i_B), $(x_B, (i+\lfloor p/2 \rfloor_B)$, (i_B, x_A), $((i+\lfloor p/2 \rfloor)_B, x_A)$, for $0 \leq i \leq \lfloor p/2 \rfloor - 1$.

If $n = 2p$ with p even, there remains uncovered the arc (x_A, x_B).

If $n = 2p+1$ with p even, there remain the 3 arcs $(x_A, x_B), (x_B, \infty)$, and (∞, x_A), which we cover with the circuit (x_A, x_B, ∞).

If $n = 2p$ with p odd, there remain the 5 arcs $(x_A, (p-1)_A), ((p-1)_A, x_B), (x_B, (p-1)_B), ((p-1)_B, x_A)$, and (x_A, x_B), which we cover with a digraph on 4 vertices containing all of them.

Finally, if $n = 2p+1$ with p odd, there remain the 7 arcs $(x_A, (p-1)_A), ((p-1)_A, x_B), (x_B, (p-1)_B), ((p-1)_B, x_A)$, $(x_A, x_B), (x_B, \infty)$, and (∞, x_A), which we cover with a digraph on 5 vertices containing all of them.

One can check that, in all cases, the arcs (u, v) considered satisfy $d_{\vec{C}_n}(u, v) \leq n/2$.

To compute the number of ADMs of this construction, we have the recurrence relations $A_{4q+2} = A_{4q} + 6q + 2$, $A_{4q+4} = A_{4q+2} + 6q + 4$, $A_{4q+3} = A_{4q+1} + 6q + 3$, and $A_{4q+5} = A_{4q+3} + 6q + 5$. Starting with $A_2 = 2$ or $A_4 = 6$ (obtained with the partition with the digraph on 4 vertices formed by the C_4 $(0, 1, 2, 3)$ plus the arc $(0, 2)$ and the digraph on 2 vertices $(1, 3)$) and $A_3 = 3$ or $A_5 = 8$ (obtained with the partition of T_5 using the first digraph on 5 vertices of Figure 3.3 and the remaining T_3), we get $A_{4q} = 6q^2 = \frac{6n^2}{16}$, $A_{4q+2} = 6q^2 + 6q + 2 = \frac{6n^2+8}{16}$, $A_{4q+1} = 6q^2 + 2q = \frac{6n^2-4n-2}{16}$, and $A_{4q+3} = 6q^2 + 8q + 3 = \frac{6n^2-4n+6}{16}$.

In all cases, the number of ADMs is of order $\frac{6}{8}\frac{n(n-1)}{2}$, so asymptotically the ratio between the number of ADMs of this construction and the lower bound of Proposition 3.4 tends to $\frac{6}{8}\frac{16}{11} = \frac{12}{11}$.

A 34/33-approximation

It will be useful to use the notation G_5 and G_6 to refer to the digraphs depicted in Figure 3.4. The key idea of this construction is that an oriented tripartite graph $K_{4,4,4}$ can be partitioned into admissible subdigraphs for $C = 2$ using 34 vertices overall, as follows.

Let the tripartition classes of the $K_{4,4,4}$ be $\{1_A, 1_B, 1_C, 1_D\}$, $\{2_A, 2_B, 2_C, 2_D\}$, $\{3_A, 3_B, 3_C, 3_D\}$, and let the vertices be ordered in the ring $1_A < 2_A < 3_A < 1_B < 2_B < 3_B < 1_C < 2_C < 3_C < 1_D < 2_D < 3_D$. The arcs of an oriented $K_{4,4,4}$ can be partitioned into 4 G_6's with $\{x_1, x_2, x_3, x_4, x_5, x_6\} = \{1_A, 2_A, 3_B, 1_C, 2_C, 3_D\}$, $\{1_B, 2_B, 3_B, 1_D, 2_D, 3_D\}$, $\{1_B, 2_C, 3_C, 1_D, 2_A, 3_A\}$, and $\{1_A, 3_A, 2_B, 1_C, 3_C, 2_D\}$, plus 2 G_5's with $\{x_1, x_2, x_3, x_4, x_5\} = \{3_A, 1_C, 2_C, 1_D, 2_D\}$ and $\{3_D, 2_A, 2_B, 1_D, 1_C\}$ (see Figure 3.4). The total number of vertices of this partition is 34.

We are now ready to explain the construction. We take an integer $p \equiv 1$ or $3 \pmod 6$, hence K_p can be partitioned into triangles. We replace each vertex i of K_p with 4 vertices

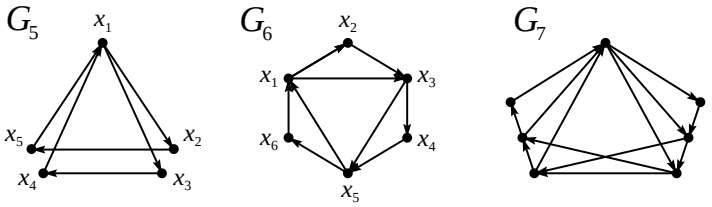

Figure 3.4: Digraphs G_5 and G_6 used in the 34/33-approximation for $C = 2$, and digraph G_7 suitable for $C = 3$ referred in the proof of Proposition 3.6.

i_A, i_B, i_C, i_D, and order the vertices $1_A < \ldots < p_A < 1_B < 2_B < \ldots < p_B < 1_C < \ldots < p_C < 1_D < \ldots < p_D$. To a triple $\{i, j, k\}$ corresponding to a triangle of K_p, with $i < j < k$, we associate the decomposition described above of the $K_{4,4,4}$ on vertices $\{\ell_A, \ell_B, \ell_C, \ell_D : \ell = i, j, k\}$. In this way, $K_{p\times 4}$ can be partitioned into $\frac{p(p-1)}{6}$ $K_{4,4,4}$'s, or equivalently into $\frac{p(p-1)}{6} \cdot 4$ G_6's and $\frac{p(p-1)}{6} \cdot 2$ G_5's. Overall, we use $\frac{34p(p-1)}{6}$ vertices. Each of the subdigraphs of this partition is admissible, as the distance in the ring between the endpoints of an arc is strictly smaller than $2p$.

To partition an oriented K_{4p}, there remain only the K_4's induced inside each class of the $K_{p\times 4}$. As $A(2, 4) = 6$, we use $6p$ vertices to cover all the K_4's.

Therefore, if $p \equiv 1$ or $3 \pmod 6$, an oriented K_{4p} can be partitioned using $6p + \frac{34p(p-1)}{6} = \frac{34p^2+2p}{6} = \frac{34n^2+8n}{96}$ vertices. To decompose K_{4p+1}, we add a vertex ∞, and we partition the p K_5's using 8 vertices for each one of them. Overall, we use $8p + \frac{34p(p-1)}{6} = \frac{34p^2+14p}{6} = \frac{34n^2-12n-24}{96}$ vertices.

If $p \not\equiv 1$ or $3 \pmod 6$, we introduce dummy vertices to get $p' \equiv 1$ or $3 \pmod 6$, we do the construction described above, and then we remove the dummy edges and vertices. It is clear that these dummy vertices add $O(n)$ vertices to the construction, hence the coefficient of the term n^2 remains the same.

Since $\frac{33n^2-24n-9}{96}$ is a lower bound by Proposition 3.4, this construction constitutes a 34/33-approximation.

3.5 Case $C = 3$

We first provide improved lower bounds for some congruence classes in Section 3.5.1 and then we provide constructions in Section 3.5.2, which are either optimal or asymptotically optimal.

3.5.1 Improved lower bounds

In this case (see Table 3.1) we have $\gamma(3, 2) = 1$, $\gamma(3, 3) = 3$, $\gamma(3, 4) = 6$, and $\gamma(3, p) = 2p$ for $p \geq 5$, so $\rho(3) = 2$. Therefore, by Theorem 3.1, we get

Proposition 3.5 $A(3,n) \geq \frac{n(n-1)}{4}$.

By Equations (3.7) and (3.9) we have

$$\begin{aligned} 2A &= \sum_{p=2}^{n} 2pa_p = 4a_2 + 6a_3 + 8a_4 + \sum_{p=5}^{n} 2pa_p \\ \frac{n(n-1)}{2} &\leq \sum_{p=2}^{n} a_p\gamma(3,p) = a_2 + 3a_3 + 6a_4 + \sum_{p=5}^{n} 2pa_p \end{aligned}$$

So,

$$A \geq \frac{n(n-1)}{4} + \frac{3}{2}a_2 + \frac{3}{2}a_3 + a_4.$$

Therefore, if the lower bound is attained, then necessarily $a_2 = a_3 = a_4 = 0$. We will see in Section 3.5.2 that this is the case for $n \equiv 1$ or $5 \pmod{12}$, using optimal digraphs on 5 vertices (namely T_5) and on 6 vertices (namely $\vec{K}_{2,2,2}$, see Figure 3.5). Optimal graphs are obtained by using arcs of length 1 and 2, so the degree of any vertex in an optimal subdigraph is 4. That is possible only if the total degree of a vertex, namely $n-1$, is a multiple of 4. Otherwise, the following proposition shows that the lower bound of Proposition 3.5 cannot be attained.

Proposition 3.6
If $n \equiv 3 \pmod 4$, $A(3,n) \geq \frac{n(n-1)}{4} + \frac{n}{6} = \frac{3n^2-n}{12}$.
If $n \equiv 0 \pmod 2$, $A(3,n) \geq \frac{n(n-1)}{4} + \frac{n}{4} = \frac{n^2}{4}$.

Proof: We use the following observation: If a vertex x has out-degree 3 (resp. in-degree 3) in a digraph B_λ, then its nearest out-neighbor A_x^+ (resp. in-neighbor A_x^-) has in-degree 1 and out-degree at most 1 (resp. out-degree 1 and in-degree at most 1). Indeed, suppose x has out-degree 3, and let A_x^+, B_x^+, C_x^+ be the out-neighbors of x. Then the load of the arc entering A_x^+ is already 3, so A_x^+ has no other in-neighbor than x. The load of the arc leaving A_x^+ is already 2, so A_x^+ has at most 1 out-neighbor y. If y has 2 or more in-neighbors, then A_x^+ is not its nearest one. Hence, to each vertex x of out-degree 3 (resp. in-degree 3) is associated a distinct vertex A_x^+ (resp. A_x^-) of degree at most 2.

Consider the digraphs in which a given vertex x appears. Let α_i^x be the number of times x appears with degree i, and let $\alpha_i = \sum_x \alpha_i^x$. Vertex x appears in $\sum_i \alpha_i^x$ digraphs, so

$$A = \sum_x \sum_i \alpha_i^x = \sum_i \alpha_i. \tag{3.18}$$

As each vertex has degree $n-1$, $n-1 = \sum_i i \cdot \alpha_i^x$, and so

$$n(n-1) = \sum_x \sum_i i \cdot \alpha_i^x = \sum_i i \cdot \alpha_i. \tag{3.19}$$

Due to the load constraint, a vertex has out-degree (resp. in-degree) at most 3 in all the digraphs in which it appears. Therefore, its degree is at most 6, that is, $\alpha_i = 0$ for $i \geq 7$. Furthermore, by the above observation if a vertex has degree 6 (resp. 5), to this vertex

are associated 2 vertices (resp. 1 vertex) of degree at most 2, and all these vertices are distinct, so

$$\alpha_1 + \alpha_2 \geq 2\alpha_6 + \alpha_5. \tag{3.20}$$

Combining Equations (3.18) and (3.19) we get

$$4A = n(n-1) + 3\alpha_1 + 2\alpha_2 + \alpha_3 - \alpha_5 - 2\alpha_6. \tag{3.21}$$

We distinguish two cases: n even or $n = 4t + 3$.

If n is even, $n - 1$ is odd and each vertex must appear at least in one B_λ with odd degree, so

$$\alpha_1 + \alpha_3 + \alpha_5 \geq n. \tag{3.22}$$

Using Equation (3.20) multiplied by 2 in Equation (3.21) we get $4A \geq n(n-1) + \alpha_1 + \alpha_3 + \alpha_5 + 2\alpha_6$, so by Equation (3.22), $4A \geq n(n-1) + n$, as claimed. Note that to obtain equality we need $\alpha_6 = 0$, $\alpha_1 + \alpha_2 = \alpha_5$, and $\alpha_1 + \alpha_3 + \alpha_5 = n$.

If $n = 4t + 3$, the degree of each vertex satisfies $n - 1 \equiv 2 \pmod 4$, so no vertex can appear with degree 4 in all the digraphs. Each vertex must appear either at least once with degree 6 or 2, or at least twice with odd degree (for example, 5 and 5, 3 and 3, 1 and 1, or 5 and 1), so

$$\alpha_2 + \alpha_6 + \frac{1}{2}(\alpha_1 + \alpha_3 + \alpha_5) \geq n. \tag{3.23}$$

Equation (3.21) can be rewritten as

$$4A = n(n-1) + \frac{2}{3}\left(\alpha_2 + \alpha_6 + \frac{1}{2}(\alpha_1 + \alpha_3 + \alpha_5)\right) + \frac{4}{3}(\alpha_2 + \alpha_1 - 2\alpha_6 - \alpha_5) + \frac{2}{3}\alpha_3 + \frac{4}{3}\alpha_1. \tag{3.24}$$

Using Equations (3.20) and (3.23) in Equation (3.24) yields $4A \geq n(n-1) + \frac{2}{3}n + \frac{2}{3}\alpha_3 + \frac{4}{3}\alpha_1$, or $A \geq \frac{n(n-1)}{4} + \frac{n}{6}$, as claimed. Note that to reach the equality, we need to have $\alpha_1 = \alpha_3 = 0$, $\alpha_2 = 2\alpha_6 + \alpha_5$ by Equation (3.20), and $2\alpha_6 + 2\alpha_2 + \alpha_5 = 2n$ by Equation (3.23), so $\alpha_2 = \frac{2n}{3}$, hence an optimal decomposition should use $\frac{n}{3}$ digraphs like the digraph G_7 depicted in Figure 3.4, having 1 vertex of degree 6 and 2 vertices of degree 2. □

3.5.2 Constructions

Our constructions rely on the existence of 3-*GDD*'s, that is, decompositions of complete multipartite graphs into K_3's. We recall the definition and some basic results below.

Decompositions or complete multipartite graphs into K_3's. Let $v_1, v_2, \ldots, v_q$ be non-negative integers; the complete multipartite graph with group sizes $v_1, v_2, \ldots, v_q$ is defined to be the graph with vertex set $V_1 \cup V_2 \cup \cdots \cup V_q$ where $|V_i| = v_i$, and two vertices $u \in V_i$ and $v \in V_j$ are adjacent if $i \neq j$. Using terminology of design theory, the graph of type $p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_h^{\alpha_h}$ is the complete multipartite graph with α_i groups of size p_i. The existence of a partition of this multipartite graph into K_k's is equivalent to the existence of a *k-GDD* (*Group Divisible Design*) of type $p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_h^{\alpha_h}$ (see [77]). Here we are interested in the existence of 3-*GDD*'s, that is, partitions into K_3's. When $|V_i| = p$ for all i, we denote by $K_{p\times q}$ the multipartite graph of type p^q. Trivial necessary conditions for the existence of a 3-*GDD* are

(i) the degree of each vertex is even; and

(ii) the number of edges is a multiple of 3.

These conditions are in general sufficient. In particular, the following results will be used later.

Theorem 3.4 ([77])
A 3-GDD of type 2^q *with* $q \geq 3$ *exists if and only if* $q \equiv 0$ *or* $1 \pmod 3$.
A 3-GDD of type $2^{q-1}4$ *with* $q \geq 4$ *exists if and only if* $q \equiv 1 \pmod 3$.
A 3-GDD of type 3^q *with* $q \geq 3$ *exists if and only if* q *is odd.*
A 3-GDD of type $3^{q-1}1$ *with* $q \geq 3$ *exists if and only if* q *is odd.*
A 3-GDD of type $3^{q-1}5$ *with* $q \geq 5$ *exists if and only if* q *is odd.*
A 3-GDD of type $3^{q-1}11$ *with* $q \geq 7$ *exists if and only if* q *is odd.*

The basic partition. In what follows $\vec{K}_{2,2,2}$ will denote the digraph on 6 vertices and 12 arcs depicted in Figure 3.5. This digraph can be viewed as being obtained from the K_3 (i, j, k) with $i < j < k$ by replacing each vertex i with two vertices i_A and i_B forming an independent set.

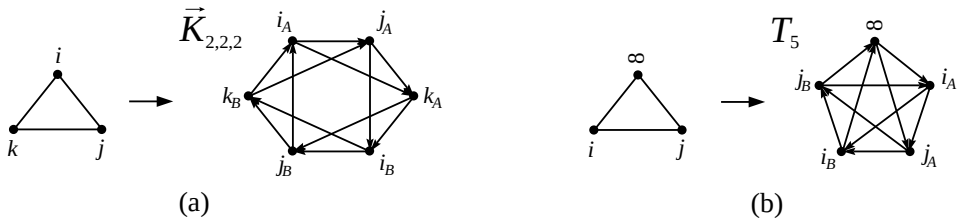

Figure 3.5: (a) Digraph $\vec{K}_{2,2,2}$ obtained from K_3 (i, j, k), with $i < j < k$; (b) digraph T_5 obtained from a K_3 of the form (∞, i, j).

Note that $\vec{K}_{2,2,2}$ is an optimal digraph for $C = 3$, since it attains the ratio $\rho(3) = 2$ (see Table 3.1). The idea of the constructions consists in starting from some graph G (mainly a multipartite graph) which can be decomposed into K_3's, replacing each vertex with two non-adjacent vertices, and then using the following lemma.

Lemma 3.1 *If a graph* $G = (V, E)$ *with vertex set* $\{1, 2, \ldots, |V|\}$ *can be decomposed into* h K_3*'s, then the digraph* H *obtained from* G *by replacing each vertex* i *with two non-adjacent vertices* i_A *and* i_B*, and where the vertices are ordered* $1_A, 2_A, \ldots, |V|_A, 1_B, 2_B, \ldots, |V|_B$*, has a valid decomposition into* $\vec{K}_{2,2,2}$*'s with a total of* $6h$ *vertices.*

Proof: To each triangle (i, j, k) with $1 \leq i < j < k \leq |V|$ is associated the $\vec{K}_{2,2,2}$ with vertices $1 \leq i_A < j_A < k_A \leq |V| < i_B < j_B < k_B \leq 2|V|$. To show that the decomposition is valid for $C = 3$, it suffices to show that the distance between the end-vertices of any arc of any $\vec{K}_{2,2,2}$ is at most $|V|$. That is true for the arcs (x_A, y_A) or (x_B, y_B) as they satisfy $x < y$, and also for the arcs (x_A, y_B) or (x_B, y_A) as they satisfy $x > y$ (see Figure 3.5(a)). □

Some small cases. We provide here decompositions of some particular small digraphs that will be used in the constructions of Propositions 3.8 and 3.9.

Lemma 3.2 *$A(3,5) = 5$, $A(3,6) \leq 10$, $A(3,7) \leq 12$, $A(3,8) \leq 18$, $A(3,9) \leq 21$, $A(3,10) \leq 28$, $A(3,11) \leq 31$, and $A(3,23) \leq 132$.*

Proof: Case $n = 5$. The decomposition is given in Figure 3.5(b), and can be viewed as obtained from the K_3 (∞, i, j) by replacing each of i, j with two vertices.

Case $n = 6, 7$. The complete graph K_4 can be decomposed into one $K_{1,3}$ $(0; \infty, 1, 2)$ and one K_3 $(\infty, 1, 2)$. Replace each of the vertices i, j, k with two vertices. The T_7 on the ordered vertices $\infty, 0_A, 1_A, 2_A, 0_B, 1_B, 2_B$ can be partitioned into a T_5 on $\infty, 1_A, 2_A, 1_B, 2_B$ ((see Figure 3.5(b) with $i = 1, j = 2$)) and the admissible digraph on 7 vertices and 11 arcs depicted in Figure 3.6(b) with $i = 0, j = 1, k = 2$. So we obtained a valid decomposition using 12 vertices. Deleting vertex ∞ yields a decomposition of T_5 with 10 vertices.

Case $n = 8, 9$. K_5 is the union of two K_3's $(\infty, 1, 3)$, $(0, 2, 3)$ and a C_4 $(\infty, 0, 1, 2)$. Replacing each vertex with two vertices we get a partition of the T_9 on the ordered vertices $\infty, 0_A, 1_A, 2_A, 3_A, 0_B, 1_B, 2_B, 3_B$. Namely, to the K_3 $(\infty, 1, 3)$ we associate a T_5 on $\infty, 1_A, 3_A, 1_B, 3_B$ (see Figure 3.5(b) with $i = 1, j = 3$). To the K_3 $(0, 2, 3)$ we associate a $\vec{K}_{2,2,2}$ on $0_A, 2_A, 3_A, 0_B, 2_B, 3_B$. To the C_4 $(\infty, 0, 1, 2)$ we associate the digraph on 7 vertices of Figure 3.6(a) with $i = 0, j = 1, k = 2$ and the triangle $(1_A, 2_A, 2_B)$. Therefore, $A(3,9) \leq 21$. Vertex 1_A appears in 3 digraphs, so $A(3,8) \leq 21 - 3 = 18$.

Case $n = 10, 11$. K_6 can be partitioned into 3 K_3's $(\infty, 1, 3), (\infty, 2, 4), (0, 1, 4)$, a star $K_{1,3}$ $(0; \infty, 2, 3)$, and a P_4 $[1, 2, 3, 4]$. Replacing each vertex with two vertices we get a partition of the T_{11} on the ordered vertices $\infty, 0_A, 1_A, 2_A, 3_A, 4_A, 0_B, 1_B, 2_B, 3_B, 4_B$ into 2 T_5's on $\infty, 1_A, 3_A, 1_B, 3_B$ and $\infty, 2_A, 4_A, 2_B, 4_B$, a $\vec{K}_{2,2,2}$ on $0_A, 1_A, 4_A, 0_B, 1_B, 4_B$, a digraph on 7 vertices and 11 arcs depicted in Figure 3.6(b) with $i = 0, j = 2, k = 3$, and an admissible digraph on 8 vertices with arcs $(1_A, 2_A), (2_A, 3_A), (3_A, 4_A), (1_B, 2_B), (2_B, 3_B), (3_B, 4_B)$, $(2_A, 1_B), (2_B, 1_A), (3_A, 2_B), (3_B, 2_A), (4_A, 3_B), (4_B, 3_A)$. Therefore, $A(3,11) \leq 31$, and as vertex ∞ appears in 3 subgraphs, we get $A(3,10) \leq 28$.

Case $n = 23$. We decompose K_{12} into 19 K_3's and 3 $K_{1,3}$'s, where vertex ∞ appears in 5 K_3's and in a star $(i; \infty, j, k)$, the two other stars being of the form $(i'; j'k', \ell')$ with $i' < j' < k' < \ell'$. We obtain a decomposition of T_{23} into 5 T_5's, 14 $\vec{K}_{2,2,2}$'s, 1 digraph of Figure 3.6(a), and 2 digraphs of Figure 3.6(c). Thus, $A(3,23) \leq 5 \cdot 5 + 14 \cdot 6 + 7 + 8 + 8 = 132$. □

Constructions. We begin with an optimal partition for $n \equiv 0, 1, 4,$ or $5 \pmod{12}$, and then we provide near-optimal constructions for the remaining values.

Proposition 3.7
If $n \equiv 0$ or $4 \pmod{12}$, $A(3,n) = \frac{n^2}{4}$.
If $n \equiv 1$ or $5 \pmod{12}$, $A(3,n) = \frac{n(n-1)}{4}$.

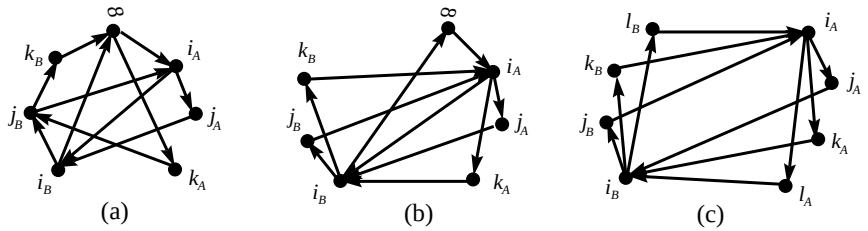

Figure 3.6: (a) Digraph associated to a C_4 (∞, i, j, k). Digraphs associated to stars ($K_{1,3}$'s), with $\infty < i < j < k < \ell$: (b) star of the form $(i; \infty, j, k)$; (c) star of the form $(i; j, k, \ell)$.

Proof: The lower bound follows from Propositions 3.5 and 3.6. For the upper bound, we will apply Lemma 3.1 with $G = K_{2\times q}$ (type 2^q), which can be decomposed by Theorem 3.4 into $\frac{2q(q-1)}{3}$ K_3's if $q \equiv 0$ or $1 \pmod 3$. As G has 2q vertices, the graph H described in Lemma 3.1 has $4q$ vertices and can be decomposed into admissible $\vec{K}_{2,2,2}$'s. Adding an admissible T_4 on each of the q independent sets of H (of the form $\{i_A, j_A, i_B, j_B\}$ where $\{i, j\}$ is an independent set of G), we get a valid decomposition of T_{4q} into q T_4's and $\frac{2q(q-1)}{3}$ admissible $\vec{K}_{2,2,2}$'s. So using $A(3,4) = 4$, we get $A(3,4q) \leq qA(3,4) + 4q(q-1) = 4q^2$ for $q \equiv 0$ or $1 \pmod 3$. So $A(3,n) \leq \frac{n^2}{4}$ for $n \equiv 0$ or $4 \pmod{12}$.

For $n = 4q + 1$, we add to the vertex set of H an extra vertex ∞. Adding to the arcs of H the q tournaments T_5 built on $\infty, i_A, j_A, i_B, j_B$, where vertices i, j are not adjacent in G, we get a decomposition of T_{4q+1} into q admissible T_5's plus $\frac{2q(q-1)}{3}$ admissible $\vec{K}_{2,2,2}$'s (the distance being at most $2q-1$ in H and so $2q$ in T_{4q+1}). Using $A(3,5) = 5$ (see Lemma 3.2), we get $A(3,4q+1) \leq qA(3,5) + 4q(q-1) = 4q^2 + q = \frac{(4q+1)4q}{4}$ for $q \equiv 0$ or $1 \pmod 3$. So $A(3,n) \leq \frac{n(n-1)}{4}$ for $n \equiv 1$ or $5 \pmod{12}$. □

We group the non-optimal constructions in Proposition 3.8 and Proposition 3.9 according to whether they differ from the lower bound by either a constant or a linear additive term, respectively.

Proposition 3.8
If $n \equiv 8 \pmod{12}$, $A(3,n) \leq \frac{n^2}{4} + 2$.
If $n \equiv 9 \pmod{12}$, $A(3,n) = \frac{n(n-1)}{4} + 3$.

Proof: We start from G of type $2^{q-1}4$ with $q \equiv 1 \pmod 3$, which can be decomposed by Lemma 3.1 into $\frac{2(q-1)(q+2)}{3}$ K_3's. As in the proof of Proposition 3.7, we get a decomposition of T_{4q+4} into $q-1$ T_4's, one T_8 and $\frac{2(q-1)(q+2)}{3}$ $\vec{K}_{2,2,2}$'s (indeed, the independent set V_q of G has 4 vertices, so in H it induces an independent set of 8 vertices). So using $A(3,4) = 4$ and $A(3,8) \leq 18$ (see Lemma 3.2), we get $A(3,4q+4) \leq (q-1)A(3,4) + A(3,8) + 4(q-1)(q+2) \leq 4q^2 + 8q + 6 = \frac{(4q+4)^2}{4} + 2$ for $q \equiv 1 \pmod 3$, so $A(3,n) \leq \frac{n^2}{4} + 2$ for $n \equiv 8 \pmod{12}$.

Similarly, adding a vertex ∞ to H we get a decomposition of T_{4q+1} into $q-1$ T_5's, one T_9 and $h = \frac{2(q-1)(q+2)}{3}$ K_3's. So using $A(3,5) = 5$ and $A(3,9) \leq 21$ we get $A(3,4q+5) \leq$

$(q-1)A(3,5)+A(3,9)+4(q-1)(q+2) \leq 4q^2+9q+8 = \frac{(4q+5)(4q+4)}{4}+3$ for $q \equiv 1 \pmod 3$, so $A(3,n) \leq \frac{n(n-1)}{4}+3$ for $n \equiv 9 \pmod{12}$. □

Proposition 3.9
If $n \equiv 2 \pmod{12}$, $A(3,n) \leq \frac{n^2}{4}+\frac{n+4}{6}$.
If $n \equiv 3 \pmod{12}$, $A(3,n) \leq \frac{n^2+3}{4}$.
If $n \equiv 6 \pmod{12}$, $A(3,n) \leq \frac{n^2}{4}+\frac{n}{6}$.
If $n \equiv 7 \pmod{12}$, $A(3,n) \leq \frac{n^2-1}{4}$.
If $n \equiv 10 \pmod{12}$, $A(3,n) \leq \frac{n^2}{4}+\frac{n+8}{6}$.
If $n \equiv 11 \pmod{12}$, $A(3,n) \leq \frac{n^2+3}{4}+\varepsilon$, *with* $\varepsilon = 1$ *for* $n = 11, 35$.

Proof: We use as graph G of Lemma 3.1 a multipartite graph of type $3^{q-1}u$ with $3(q-1)+u$ vertices, in order to get a decomposition of $T_{6(q-1)+2u}$ (resp. $T_{6(q-1)+2u+1}$) into $q-1$ T_6's (resp. T_7's), one T_{2u} (resp. T_{2u+1}) and the digraph H itself decomposed by Lemma 3.1 into $h = \frac{9(q-1)(q-2)}{6}+u(q-1)$ $\vec{K}_{2,2,2}$'s. We distinguish several cases according to the value of u.

Case 1: $u = 1$, $q \geq 3$ odd.

Let $n \equiv 2 \pmod{12}$, $n = 6q-4$. Using that $A(3,2) = 2$ and $A(3,6) \leq 10$ we get $A(3,6q-4) \leq (q-1)A(3,6)+A(3,2)+(q-1)(9q-12) \leq 9q^2-11q+4 = \frac{(6q-4)^2}{4}+q = \frac{n^2}{4}+\frac{n+4}{6}$.

Let $n \equiv 3 \pmod{12}$, $n = 6q-3$. Using that $A(3,3) = 3$ and $A(3,7) \leq 12$ we get $A(3,6q-3) \leq (q-1)A(3,7)+A(3,3)+(q-1)(9q-12) \leq 9q^2-9q+3 = \frac{(6q-3)^2}{4}+\frac{3}{4} = \frac{n^2+3}{4}$.

Case 2: $u = 3$, $q \geq 3$ odd.

Let $n \equiv 6 \pmod{12}$, $n = 6q$. Using that $A(3,6) \leq 10$ we get $A(3,6q) \leq qA(3,6)+9q(q-1) \leq 9q^2+q = \frac{n^2}{4}+\frac{n}{6}$.

Let $n \equiv 7 \pmod{12}$, $n = 6q+1$. Using that $A(3,7) \leq 12$ we get $A(3,6q+1) \leq qA(3,7)+9q(q-1) \leq 9q^2+3q = \frac{n^2-1}{4}$.

Case 3: $u = 5$, $q \geq 5$ odd.

Let $n \equiv 10 \pmod{12}$, $n = 6q+4$. Using that $A(3,6) \leq 10$ and $A(3,10) \leq 28$ we get $A(3,6q+4) \leq (q-1)A(3,6)+A(3,10)+(q-1)(9q+12) \leq 9q^2+13q+6 = \frac{(6q+4)^2}{4}+\frac{6q+12}{6} = \frac{n^2}{4}+\frac{n+8}{6}$.

Let $n \equiv 11 \pmod{12}$, $n = 6q+5$. Using that $A(3,7) \leq 12$ and $A(3,11) \leq 31$ we get $A(3,6q+5) \leq (q-1)A(3,7)+A(3,11)+(q-1)(9q+12) \leq 9q^2+15q+7 = \frac{n^2+3}{4}$.

For $q = 23$ we have $A(3,23) \leq 132 = \frac{23^2-1}{4}$, one less than the value given by the preceding construction. Using $u = 11$, $q \geq 7$ odd, $n = 6q+17$, $A(3,7) \leq 12$, and $A(3,23) \leq 132$ we get $A(3,6q+17) \leq (q-1)A(3,7)+A(3,23)+(q-1)(9q+48) \leq 9q^2+51q+72 = \frac{(6q+17)^2-1}{4} = \frac{n^2-1}{4}$. It might be that $A(3,11) \leq 30$, and then the bound $\frac{n^2-1}{4}$ would be also attained for $n = 11$ and 35. □

3.6 Case $C > 3$

For $C > 3$, we distinguish two cases according to whether C is of the form $\frac{k(k+1)}{2}$ or not. We focus on those cases in Sections 3.6.1 and 3.6.2.

3.6.1 C not of the form $k(k+1)/2$

If C is not of the form $\frac{k(k+1)}{2}$, we can improve the lower bound of Theorem 3.1, as we did for $C = 2$ in Proposition 3.4. We provide the details for $C = 4$ and sketch the ideas for $C = 5$, that show how to improve the lower bound for any value of C not of the form $k(k+1)/2$.

Proposition 3.10

$$A(4,n) \;\geq\; \frac{7}{32}n(n-1) \;=\; \left(\frac{3}{14}+\frac{1}{224}\right)n(n-1).$$

Proof: The values of $\gamma(4,p)$ are given in Table 3.1, so Equation (3.11) becomes in the case $C = 4$

$$A = \sum_{p=2}^{n} pa_p \;\geq\; \frac{3}{7}\sum_{p=2}^{n} a_p\gamma(4,p)+\frac{11}{7}a_2+\frac{12}{7}a_3+\frac{10}{7}a_4+\frac{5}{7}a_5+\frac{3}{7}a_6+\frac{1}{7}(a_7+2a_8+a_{10}+2a_{11}+a_{13}+2a_{14}+\ldots). \tag{3.25}$$

Using that $\sum_{p=2}^{n} a_p\gamma(4,p) \geq \frac{n(n-1)}{2}$, Equation (3.25) becomes

$$14A \;\geq\; 3n(n-1)+22a_2+24a_3+20a_4+10a_5+6a_6+2a_7+4a_8+\ldots \tag{3.26}$$

On the other hand,

$$A \;\geq\; 9\left(\Lambda-\sum_{i=2}^{8} a_i\right)+\sum_{i=2}^{8} i\cdot a_i \;=\; 9\Lambda-7a_2-6a_3-5a_4-4a_5-3a_6-2a_7-a_8. \tag{3.27}$$

Summing Equations (3.26) and (3.27) and using that $\Lambda \geq \frac{n(n-1)}{32}+\frac{n-1}{32}$ by Proposition 3.1 yields

$$15A \;\geq\; \frac{105}{32}n(n-1)+\frac{9}{32}(n-1), \quad \text{and therefore} \quad A \;\geq\; \frac{7}{32}n(n-1)+\frac{3}{160}(n-1).$$

□

For $C = 5$, a similar computation with $\rho(5) = 8/3$ gives

$$8A \;\geq\; \frac{3}{2}n(n-1)+13a_2+15a_3+14a_4+10a_5+3a_6+2a_7+a_8. \tag{3.28}$$

$$A \;\geq\; 9\Lambda-7a_2-6a_3-5a_4-4a_5-3a_6-2a_7-a_8. \tag{3.29}$$

So again, Summing Equations (3.28) and (3.29) and using that $\Lambda \geq \frac{n(n-1)}{40}+\frac{n-1}{40}$ by Proposition 3.1 yields

$$A \;\geq\; \frac{n(n-1)}{6}+\frac{n(n-1)}{40}+\frac{n-1}{40} \;=\; \frac{23}{120}n(n-1)+\frac{n-1}{40} \;=\; \left(\frac{3}{16}+\frac{1}{240}\right)n(n-1)+\frac{n-1}{40}.$$

3.6.2 C of the form $k(k+1)/2$

For $C = \frac{k(k+1)}{2}$ the lower bound of Theorem 3.1 can be attained, according to the existence of a type of *k-GDD*, called *Balanced Incomplete Block Design* (*BIBD*). A $(v, k, 1)$-*BIBD* consists simply of a partition of K_v into K_k's.

Theorem 3.5 *If there exists a $(k+1)$-GDD of type k^q (that is, a decomposition of $K_{k\times q}$ into K_{k+1}'s), then there exists an optimal admissible partition of T_{2kq+1} for $C = \frac{k(k+1)}{2}$ with $\frac{n(n-1)}{2k}$ ADMs.*

Proof: The lower bounds follows from Theorem 3.1. For the upper bound, as we did in Proposition 3.7 (case $k = 2$, $C = 2$), we replace each vertex i of $K_{k\times q}$ with two vertices i_A and i_B, and add a new vertex ∞. We label the vertices of the obtained T_{2kq+1} with $\infty, 1_A, \ldots, (kq)_A, 1_B, \ldots, (kq)_B$. To each K_{k+1} of the decomposition of $K_{k\times q}$ we associate a $T_{2\times(k+1)}$, which is an optimal digraph for $C = \frac{k(k+1)}{2}$ with $2(k+1)$ vertices and $2k(k+1)$ edges, hence attaining $\rho(C) = k$. So adding vertex ∞ to the stable sets of size $2k$ we obtain a decomposition of T_{2kq+1} into q T_{2k+1}'s (which are also optimal) and $T_{2\times(k+1)}$'s.

If $K_{k\times q}$ is decomposable into K_{k+1}'s, the number of K_{k+1}'s (and so the number of $T_{2\times(k+1)}$'s) is $\frac{kq(q-1)}{k+1}$. Therefore the total number of ADMs is $q(2k+1) + 2kq(q-1) = \frac{(2kq+1)2kq}{2k} = \frac{n(n-1)}{2k}$. □

Note that a decomposition of $K_{k\times q}$ into K_{k+1}'s is equivalent to a decomposition of K_{kq+1} into K_{k+1}'s by adding a new vertex ∞, that is, a $(kq+1, k+1, 1)$-*BIBD*. In particular, such designs are known to exist if n is large enough and $(kq+1)kq \equiv 0 \pmod{k(k+1)}$ [77]. For example, for $k = 3$ and $q \equiv 0$ or $1 \pmod 4$, or $k = 4$ and $q \equiv 0$ or $1 \pmod 5$.

Corollary 3.1
If $C = 6$ and $n \equiv 1$ or $7 \pmod{24}$, $A(6, n) = \frac{n(n-1)}{6}$.
If $C = 10$ and $n \equiv 1$ or $9 \pmod{40}$, $A(10, n) = \frac{n(n-1)}{8}$.

Corollary 3.2 *For $C \in \{15, 21, 28, 36\}$, there exists a small set of values of n for which the existence of a BIBD remains undecided (179 values overall, see [77, pages 73-74]). For the values of n different from these undecided BIBDs, the following results apply.*
If $C = 15$ and $n \equiv 1$ or $11 \pmod{30}$, $A(15, n) = \frac{n(n-1)}{10}$.
If $C = 21$ and $n \equiv 1$ or $13 \pmod{84}$, $A(21, n) = \frac{n(n-1)}{12}$.
If $C = 28$ and $n \equiv 1$ or $15 \pmod{112}$, $A(28, n) = \frac{n(n-1)}{14}$.
If $C = 36$ and $n \equiv 1$ or $17 \pmod{144}$, $A(36, n) = \frac{n(n-1)}{16}$.

Wilson proved [205] that for v large enough, K_v can be decomposed into subgraphs isomorphic to any given graph G, if the trivial necessary conditions about the degree and the number of edges are satisfied. Thus, we can assure that optimal constructions exist when $C = \frac{k(k+1)}{2}$ for all $k > 0$.

Corollary 3.3 *If $C = \frac{k(k+1)}{2}$, then $A(C, n) = \frac{n(n-1)}{2k}$ for $n \equiv 1$ or $2k+1 \pmod{4C}$ large enough.*

We can also use decompositions of $K_{p\times q}$ into K_{k+1}'s to get constructions asymptotically optimal, but not attaining the lower bound like for $C = 3$. For instance, for $C = 6$ the proof of Theorem 3.5 gives (without adding the vertex ∞) that for $q \equiv 0$ or $1 \pmod 4$ and $n \equiv 0$ or $6 \pmod{24}$,

$$A(6, 6q) \;\leq\; qA(6, 6) + 6q(q-1) \;=\; 6q^2 \;=\; \frac{n^2}{6}.$$

That might be an optimal value if we could improve the lower bound for $C = 6$ as we did for $C = 3$ in Proposition 3.6, but the calculations become considerably more complicated.

Corollary 3.4
For $n \equiv 0$ *or* $6 \pmod{24}$, $\frac{n(n-1)}{6} \;\leq\; A(6, n) \;\leq\; \frac{n^2}{6}$.
For $n \equiv 0$ *or* $8 \pmod{40}$, $\frac{n(n-1)}{8} \;\leq\; A(10, n) \;\leq\; \frac{n^2}{8}$.

For a general C of the form $C = \frac{k(k+1)}{2}$, the improved lower bound one could expect is $\frac{n^2}{2k}$.

3.7 Unidirectional or Bidirectional Rings?

This section is devoted to compare unidirectional and bidirectional rings in terms of minimizing electronics cost, when these rings are used in a WDM network with traffic grooming. In [48] general lower bounds are given in unidirectional rings:

$$A_{uni}(C, n) \geq \frac{n(n-1)}{2}\frac{1}{\eta(C)},$$

$$\text{where } \eta(C) = \begin{cases} \frac{k-1}{2}, & \text{if } \frac{(k-1)k}{2} \leq C \leq \frac{(k-1)(k+1)}{2} \\ \frac{C}{k+1}, & \text{if } \frac{(k-1)(k+1)}{2} \leq C \leq \frac{k(k+1)}{2} \end{cases}$$

In this chapter we have provided a general lower bound for all values of C and n in bidirectional rings (taking into account requests both clockwise and counterclockwise):

$$A_{bi}(C, n) \geq \frac{n(n-1)}{2}\frac{2k}{(k-1)k + r},$$

where $C = 1 + 2 + \ldots + k - 1 + r = \frac{(k-1)k}{2} + r$, with $0 \leq r < k$.

The first observation is that both bounds behave like $\binom{n}{2}$ on n, and like $\frac{1}{\sqrt{C}}$ on C. Furthermore, the bounds coincide when $C = (k-1)k/2$ for all $k > 0$. It is straightforward to prove that for all other values of C the unidirectional ring bound is strictly larger. This is what one can expect a priori, because the set of possible routings in a bidirectional ring contains unidirectional routing as a particular case. In this chapter we have focused on symmetric routing following a shortest path in the bidirectional ring, so this result gives indeed important information. We depict both bounds in the first graph of Figure 3.7. In this figure the bounds are normalized by $n(n-1)/2$ and the horizontal axis is logarithmic

on C. The solid (resp. dashed) line represents the bidirectional (resp. unidirectional) bound.

A natural question is the following: does it exist a construction for bidirectional rings with cost strictly smaller that the lower bound for unidirectional rings? Indeed, consider $C = 2$, and then the ratio between both bounds is $12/11 = 1,0909$. That is, if we have an α-approximation for bidirectional rings with $\alpha < 12/11$, the cost of this solution is strictly smaller than the lower bound for unidirectional rings. The 34/33-approximation provided in Section 3.4.2 is such an example. Thus, we conclude that the cost in bidirectional rings with shortest path routing, in terms of number of ADMs needed, can be strictly smaller than the cost in unidirectional rings.

Because of Wilson's theorem [205], both lower bounds are achieved for infinite values of C and n. Thus, it makes sense to compare these bounds to infer which type of routing is better in terms of electronics cost. Let us consider the ratio between both bounds, namely $A^*_{uni}(C,n)$ and $A^*_{bi}(C,n)$. If $\frac{(k-1)k}{2} \leq C \leq \frac{(k-1)(k+1)}{2}$:

$$1 \leq \frac{A^*_{uni}(C,n)}{A^*_{bi}(C,n)} = \frac{(k-1)k+r}{(k-1)k} = 1 + \frac{r}{(k-1)k} \leq 1 + \frac{1}{k}$$

Finally, if $\frac{(k-1)(k+1)}{2} \leq C \leq \frac{k(k+1)}{2}$:

$$1 \leq \frac{A^*_{uni}(C,n)}{A^*_{bi}(C,n)} = \frac{k+1}{C}\frac{(k-1)k+r}{2k} = \left(1+\frac{1}{k}\right)\left(1-\frac{r}{2C}\right) \leq 1 + \frac{1}{k}$$

That is, in all cases we have that

$$1 \leq \frac{A^*_{uni}(C,n)}{A^*_{bi}(C,n)} \leq 1 + \frac{1}{k} \tag{3.30}$$

From Equation (3.30) we conclude that

$$\lim_{C\to\infty} \frac{A^*_{uni}(C,n)}{A^*_{bi}(C,n)} = 1, \qquad \forall\ n > 0,$$

as we can see in the second graph of Figure 3.7.

Hence, for big values of n there is no real improvement in bidirectional rings, in terms of cost of electronic switching. Is this conclusion surprising? In fact, our objective function it the number of electronic terminations, that is, the number of ADMs. And, roughly speaking, we need to place an ADM in a node when a request either originates or terminates, regardless of routing. Thus, it is not that surprising that, asymptotically, bidirectional routing does not improve unidirectional routing. Of course, there is a drawback when using unidirectional rings. This drawback is the bandwidth utilization. Let us say that a request routed along a path of length l uses l *units of bandwidth*. In the all-to-all case, in a unidirectional ring on n nodes each pair of communicating nodes uses n units of bandwidth, since the requests (i,j) and (j,i) are routed via disjoint paths. Thus, the total bandwidth utilization is $\binom{n}{2} \cdot n$, and the load of each link is $\frac{n(n-1)}{2}$. On the other hand, in a bidirectional ring it is easy to compute the load of a link, that turns out to be $\frac{n^2-1}{4}$ for n odd. That is, the bandwidth utilization is asymptotically twice better in bidirectional rings. In conclusion, there is a trade-off between bandwidth utilization (better in bidirectional rings) and technological simplicity (better in unidirectional rings). But, concerning the electronics cost, both rings behave in a similar fashion.

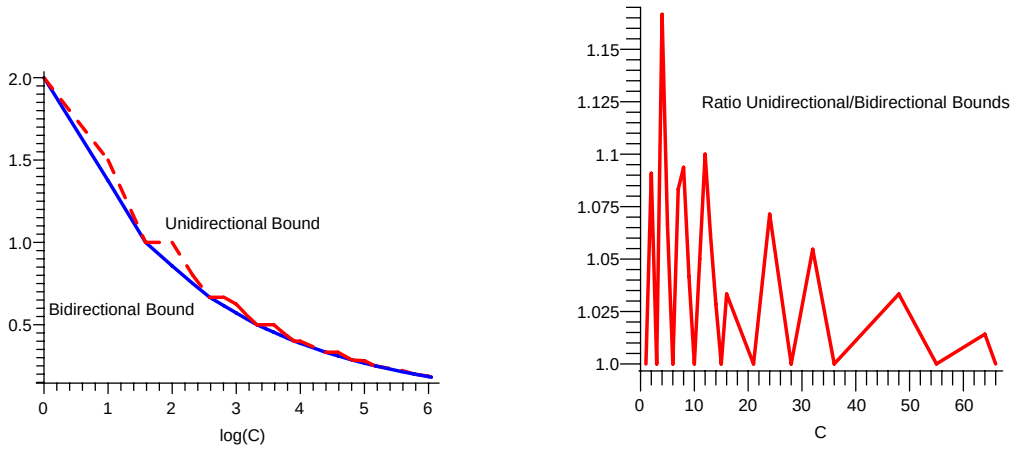

Figure 3.7: Comparison of lower bounds for unidirectional and bidirectional rings.

3.8 Conclusions

In this chapter we studied the minimization of ADMs in optical WDM bidirectional ring networks under the assumption of symmetric shortest path routing and all-to-all unitary requests. We precisely formulated the problem in terms of graph decompositions, and stated a general lower bound for all the values of C and n. We then studied extensively the cases $C = 2$ and $C = 3$, providing improved lower bounds, optimal constructions for several infinite families, as well as asymptotically optimal constructions and approximations. To the best of our knowledge, these are the first optimal solutions in the literature for traffic grooming in bidirectional rings. We then study the case $C > 3$, focusing specifically on the case $C = k(k+1)/2$ for some $k \geq 1$. We gave optimal decompositions for several congruence classes of n, using the existence of a certain combinatorial design. We concluded with a comparison of the switching cost in unidirectional and bidirectional WDM rings.

Further research is needed to find new families of optimal solutions for other values of C. The first step should be to improve the general lower bound for other values of C, namely, finding a closed formula. It would be interesting to consider other kinds of routing in bidirectional rings, not necessarily symmetric or using shortest paths. Stating which kind of routing is the best for each value of n and C would be a nice result. Finally, studying the traffic grooming problem using graph partitioning tools in other topologies, like trees or hypercubes, would be also interesting.

Chapter 4

Two-period Grooming

In this chapter we study grooming for two-period optical networks, a variation of the traffic grooming problem for WDM ring networks introduced by Colbourn, Quattrocchi, and Syrotiuk [80, 81]. In the two-period grooming problem, during the first period of time, there is all-to-all uniform traffic among n nodes, each request using $1/C$ of the bandwidth; and during the second period, there is all-to-all uniform traffic only among a subset V of v nodes, each request now being allowed to use $1/C'$ of the bandwidth, where $C' < C$. We determine the minimum drop cost (minimum number of ADMs) for any n, v and $C = 4$ and $C' \in \{1, 2, 3\}$. To do this, we use tools of graph decompositions. Indeed the two-period grooming problem corresponds to minimizing the total number of vertices in a partition of the edges of the complete graph K_n into subgraphs, where each subgraph has at most C edges and where furthermore it contains at most C' edges of the complete graph on v specified vertices. Subject to the condition that the two-period grooming has the least drop cost, the minimum number of wavelengths required is also determined in each case.

Keywords: traffic grooming, SONET ADM, optical networks, graph decomposition, design theory.

4.1 Introduction

In this chapter we deal with the traffic grooming problem for a unidirectional SONET ring with n nodes, grooming ratio C, and all-to-all uniform unitary traffic. This problem has been modeled as a graph partition problem in both [48] and [138]. In the all-to-all case the set of requests is modeled by the complete graph K_n. To a wavelength λ is associated a subgraph B_λ in which each edge corresponds to a pair of symmetric requests (that is, a circle) and each node to an ADM. The grooming constraint, i.e., the fact that a wavelength can carry at most C requests, corresponds to the fact that the number of edges $|E(B_\lambda)|$ of each subgraph B_λ is at most C. The cost corresponds to the total number of vertices used in the subgraphs, and the objective is therefore to minimize this number.

Traffic Grooming in Unidirectional Rings with All-to-all Traffic
Input: Two integers n and C.
Output: Partition $E(K_n)$ into subgraphs B_λ, $1 \le \lambda \le \Lambda$, s.t. $|E(B_\lambda)| \le C$ for all λ.
Objective: Minimize $\sum_{\lambda=1}^{\Lambda} |V(B_\lambda)|$.

With the all-to-all set of requests, optimal constructions for a given grooming ratio C have been obtained using tools of graph and design theory [77]. See Section II.3.2 (page 36) for a survey of the results in unidirectional rings with all-to-all traffic. Graph decompositions have been extensively studied for other reasons as well. See [62] for an excellent survey, [83] for relevant material on designs with blocksize three, and [77] for terminology in design theory.

Most of the papers on grooming deal with a single (static) traffic matrix. Some articles consider variable (dynamic) traffic, such as finding a solution which works for the maximum traffic demand [56, 210] or for all request graphs with a given maximum degree [C15, C18], but all keep a fixed grooming factor. In [81] an interesting variation of the traffic grooming problem, grooming for two-period optical networks, has been introduced in order to capture some dynamic nature of the traffic. Informally, in the two-period grooming problem each time period supports different traffic requirements. During the first period of time there is all-to-all uniform traffic among n nodes, each request using $1/C$ of the bandwidth; but during the second period there is all-to-all traffic only among a subset V of v nodes, each request now being allowed to use a larger fraction of the bandwidth, namely $1/C'$ where $C' < C$.

Denote by X the subset of n nodes. Therefore the two-period grooming problem can be expressed as follows.

Two-Period Grooming in the Ring
Input: Four integers n, v, C, and C'.
Output: A partition (denoted $N(n, v; C, C')$) of $E(K_n)$ into subgraphs B_k, $1 \le \lambda \le \Lambda$, such that for all λ, $|E(B_\lambda)| \le C$, and $|E(B_\lambda) \cap (V \times V)| \le C'$, with $V \subseteq X$, $|V| = v$.
Objective: Minimize $\sum_{\lambda=1}^{\Lambda} |V(B_\lambda)|$.

Following [80], a grooming is denoted by $N(n, C)$. When the grooming $N(n, C)$ is *optimal*, i.e., minimizes the total ADM cost, then the grooming is denoted by $\mathscr{ON}(n, C)$. Whether general or optimal, the drop cost of a grooming is denoted by *cost* $N(n, C)$ or *cost* $\mathscr{ON}(n, C)$, respectively.

A grooming of a two-period network $N(n, v; C, C')$ with grooming ratios (C, C') coincides with a graph decomposition $(X, \mathcal{B})$ of K_n (using standard design theory terminology, $\mathcal{B}$ is the set of all the *blocks* of the decomposition) such that $(X, \mathcal{B})$ is a grooming $N(n, C)$ in the first time period, and $(X, \mathcal{B})$ faithfully embeds a graph decomposition of K_v such that $(V, \mathcal{D})$ is a grooming $N(v, C')$ in the second time period. Let $V \subseteq X$. The graph decomposition $(X, \mathcal{B})$ *embeds* the graph decomposition $(V, \mathcal{D})$ if there is a mapping $f : \mathcal{D} \to \mathcal{B}$ such that D is a subgraph of $f(D)$ for every $D \in \mathcal{D}$. If f is injective (i.e., one-to-one), then $(X, \mathcal{B})$ *faithfully embeds* $(V, \mathcal{D})$. This concept of faithfully embedding has been explored in [79, 180].

We use the notation $\mathscr{ON}(n, v; C, C')$ to denote an optimal grooming $N(n, v; C, C')$.

As it turns out, an $\mathscr{ON}(n, v; C, C')$ does not always coincide with an $\mathscr{ON}(n, C)$. Generally we have *cost* $\mathscr{ON}(n, v; C, C') \geq$ *cost* $\mathscr{ON}(n, C)$ (see Examples 4.2 and 4.3). Of particular interest is the case when *cost* $\mathscr{ON}(n, v; C, C') =$ *cost* $\mathscr{ON}(n, C)$ (see Example 4.1).

Example 4.1 *Let $n = 7$, $v = 4$, $C = 4$. Let $V = \{0, 1, 2, 3\}$ and $W = \{a_0, a_1, a_2\}$. An optimal decomposition is given by the three triangles $(a_0, 0, 1)$, $(a_1, 1, 2)$, and $(a_2, 2, 3)$, and the three 4-cycles $(0, 2, a_0, a_1)$, $(0, 3, a_0, a_2)$, and $(1, 3, a_1, a_2)$, giving a total cost of 21 ADMs.*

This solution is valid and optimal for both $C' = 1$ and $C' = 2$, and it is optimal for the classical Traffic Grooming in the Ring *problem when $n = 7$ and $C = 4$. Therefore, cost $\mathscr{ON}(7, 4; 4, 1) =$ cost $\mathscr{ON}(7, 4; 4, 2) =$ cost $\mathscr{ON}(7, 4) = 21$.*

Example 4.2 *Let $n = 7$, $v = 5$, $C = 4$, and $C' = 2$. Let $V = \{0, 1, 2, 3, 4\}$ and $W = \{a_0, a_1\}$. We see later that an optimal decomposition is given by the five kites $(a_0, 1, 2; 0)$, $(a_0, 3, 4; 1)$, $(a_1, 1, 3; 2)$, $(a_1, 2, 4; 0)$ and $(a_0, a_1, 0; 1)$, plus the edge $\{0, 3\}$, giving a total cost of 22 ADMs. So cost $\mathscr{ON}(7, 5; 4, 2) = 22$. Note that this decomposition is not a valid solution for $C' = 1$, since there are subgraphs containing more than one edge with both end-vertices in V.*

Example 4.3 *Let $n = 7$, $v = 5$, $C = 4$, and $C' = 1$. Let again $V = \{0, 1, 2, 3, 4\}$ and $W = \{a_0, a_1\}$. We see later that an optimal decomposition is given by the four K_3s $(a_0, 1, 2)$, $(a_0, 3, 4)$, $(a_1, 0, 3)$, and $(a_1, 2, 4)$, the C_4 $(0, 1, a_1, a_0)$, plus the five edges $\{0, 4\}$, $\{1, 3\}$, $\{0, 2\}$, $\{1, 4\}$, and $\{2, 3\}$, giving a total cost of 26 ADMs. So cost $\mathscr{ON}(7, 5; 4, 1) = 26$.*

Colbourn, Quattrocchi, and Syrotiuk [80, 81] completely solved the cases when $C = 2$ and $C = 3$ ($C' = 1$ or 2). In this chapter we determine the minimum drop cost of an $N(n, v; 4, C')$ for all $n \geq v \geq 0$ and $C' \in \{1, 2, 3\}$.

We are also interested in determining the minimum number of wavelengths, or *wavecost*, required in an assignment of wavelengths to a decomposition. Among the $\mathscr{ON}(n, 4)$s one having the minimum wavecost is denoted by $\mathscr{MON}(n, 4)$, and the corresponding minimum number of wavelengths by *wavecost*$\mathscr{MON}(n, 4)$. We characterize the $\mathscr{ON}(n, v; C, C')$ whose wavecost is minimum among all $\mathscr{ON}(n, v; C, C')$s, and which is denoted by $\mathscr{MON}(n, v; C, C')$; the wavecost is itself denoted by *wavecost*$\mathscr{MON}(n, v; C, C')$.

We deal separately with each value of $C' \in \{1, 2, 3\}$. Table 4.1 summarizes the cost formulas for $n = v + w > 4$.

4.2 Preliminaries

We establish some notation to be used throughout the chapter. K_n denotes a complete graph on n vertices and K_X represents the complete graph on the vertex set X. A triangle with edges $\{\{x, y\}, \{x, z\}, \{y, z\}\}$ is denoted by (x, y, z). A 4-cycle with edges $\{\{x, y\}, \{y, z\}, \{z, u\}, \{u, x\}\}$ is denoted by (x, y, z, u). A *kite* with edges $\{\{x, y\}, \{x, z\}, \{y, z\}, \{z, u\}\}$ is denoted by $(x, y, z; u)$. The groomings to be produced also employ paths; the path on k

$$cost\ \mathscr{ON}(v+w, v; 4, 1) = \begin{cases} \binom{v+w}{2} & \text{if } v \le w+1 \\ \binom{v+w}{2} + \binom{v}{2} - \left\lfloor \frac{vw}{2} \right\rfloor & \text{if } v \ge w+1 \end{cases}$$

$$cost\ \mathscr{ON}(v+w, v; 4, 2) = \begin{cases} \binom{v+w}{2} & \text{if } v \le 2w \\ \binom{v+w}{2} + \left\lceil \frac{1}{2}\binom{v}{2} \right\rceil - \frac{vw}{2} + \delta & \text{if } v > 2w \text{ and } v \text{ even} \\ \quad \text{where } \delta = \begin{cases} 1 & \text{if } w = 2, \text{ or} \\ & \text{if } w = 4 \text{ and} \\ & \quad v \equiv 0 \pmod 4 \\ 0 & \text{otherwise} \end{cases} & \\ \binom{v+w}{2} + \left\lceil \frac{1}{2}\left(\binom{v}{2} - vw - \left\lceil \frac{w}{2} \right\rceil\right) \right\rceil + \delta & \text{if } v > 2w \text{ and } v \text{ odd} \\ \quad \text{where } \delta = \begin{cases} 1 & \text{if } w = 3 \text{ and} \\ & \quad v \equiv 3 \pmod 4 \\ 0 & \text{otherwise} \end{cases} & \end{cases}$$

$$cost\ \mathscr{ON}(v+w, v; 4, 3) = \binom{v+w}{2}$$

Table 4.1: Cost formulas for $n = v + w > 4$.

vertices P_k is denoted by $[x_1, \ldots, x_k]$ when it contains edges $\{x_i, x_{i+1}\}$ for $1 \le i < k$. Now let $G = (X, E)$ be a graph. If $|X|$ is even, a set of $|X|/2$ disjoint edges in E is a *1-factor*; a partition of E into 1-factors is a *1-factorization*. Similarly, if $|X|$ is odd, a set of $(|X|-1)/2$ disjoint edges in E is a *near 1-factor*; a partition of E into near 1-factors is a *near 1-factorization*. We also employ well-known results on partial triple systems and group divisible designs with block size three; see [83] for background.

The vertices of the set V are the integers modulo v denoted by $0, 1, \ldots, v-1$. The vertices not in V, that is in $X \setminus V$, forms the set W of size $w = n - v$ and is denoted by $a_0, \ldots, a_{w-1}$, the indices being taken modulo w.

Among graphs with three or fewer edges (i.e., when $C = 3$),the only graph with the minimum ratio (number of vertices over the number of edges) is the triangle. For $C = 4$ three different such graphs have minimum ratio 1: the triangle, the 4-cycle, and the kite. This simplifies the problem substantially. Indeed, in contrast to the lower bounds in [81], in this case the lower bounds arise from easy classification of the edges on V. We recall the complete characterization for optimal groomings with a grooming ratio of four:

Theorem 4.1 [48, 151] *cost* $\mathscr{ONN}(4,4) = 7$ *and, for* $n \ge 5$, *cost* $\mathscr{ONN}(n,4) = \binom{n}{2}$. *Furthermore a* $\mathscr{MONN}(4,4)$ *employs two wavelengths and can be realized by a kite and a* P_3 *(or a* K_3 *and a star), and a* $\mathscr{MONN}(n,4)$, $n \ge 5$, *employs* $\left\lceil \frac{n(n-1)}{8} \right\rceil$ *wavelengths and can be realized by* t K_3*s and* $\left\lceil \frac{n(n-1)}{8} - t \right\rceil$ *4-cycles or kites, where*

$$t = \begin{cases} 0 & \text{if } n \equiv 0,1 \pmod 8 \\ 1 & \text{if } n \equiv 3,6 \pmod 8 \\ 2 & \text{if } n \equiv 4,5 \pmod 8 \\ 3 & \text{if } n \equiv 2,7 \pmod 8 \end{cases}.$$

In order to unify the treatment of the lower bounds, in a decomposition $N(v+w, v; 4, C')$ for $C' \in \{1, 2\}$, we call an edge with both ends in V *neutral* if it appears in a triangle, 4-cycle, or kite; we call it *positive* otherwise. An edge with one end in V and one in W is a *cross edge*.

Lemma 4.1

1. *In an* $N(v+w, v; 4, C')$ *with* $C' \in \{1, 2\}$, *the number of neutral edges is at most* $\frac{1}{2}C'vw$.
2. *When* v *is odd and* $C' = 2$, *the number of neutral edges is at most* $vw - \frac{w}{2}$.

Proof: Every neutral edge appears in a subgraph having at least two cross edges. Thus the number of subgraphs containing one or more neutral edges is at most $\frac{1}{2}vw$. Each can contain at most C' neutral edges, and hence there are at most $\frac{1}{2}C'vw$ neutral edges. This proves the first statement.

Suppose now that $C' = 2$ and v is odd. Any subgraph containing two neutral edges employs exactly two cross edges incident to the same vertex in W. Thus the number α of such subgraphs is at most $\frac{1}{2}w(v-1)$. Then remaining neutral edges must arise (if present) in triangles, kites, or 4-cycles that again contain two cross edges but only one neutral edge; their number, β, must satisfy $\beta \le \frac{vw}{2} - \alpha$. Therefore the number of neutral edges, $2\alpha + \beta$, satisfies $2\alpha + \beta \le \frac{1}{2}w(v-1) + \frac{vw}{2} = vw - \frac{w}{2}$. □

When $C = 3$ there are strong interactions among the decompositions placed on V, on W, and on the cross edges [80,81]; fortunately here we shall see that the structure on V suffices to determine the lower bounds. Because every $N(v+w, v; 4, C')$ is an $N(v+w, v; 4, C'+1)$ for $1 \le C' \le 3$, and $N(v+w, v; 4, 4)$ coincides with $N(v+w, 4)$, $cost\ \mathscr{ON}(v+w, v; 4, 1) \ge cost\ \mathscr{ON}(v+w, v; 4, 2) \ge cost\ \mathscr{ON}(v+w, v; 4, 3) \ge cost\ \mathscr{ON}(v+w, 4)$. We use these obvious facts to establish lower and upper bounds without further comment.

4.3 Case $C' = 1$

4.3.1 $\mathscr{ON}(n, v; 4, 1)$

Theorem 4.2 *Let $n = v + w \ge 5$.*

1. *$cost\ \mathscr{ON}(v+w, v; 4, 1) = cost\ \mathscr{ON}(v+w, 4)$ when $v \le w+1$.*
2. *$cost\ \mathscr{ON}(v+w, v; 4, 1) = \binom{v+w}{2} + \binom{v}{2} - \lfloor \frac{vw}{2} \rfloor$ when $v \ge w+1$.*

Proof: To prove the lower bound, we establish that $cost\ \mathscr{ON}(v+w, v; 4, 1) \ge \binom{v+w}{2} + \binom{v}{2} - \lfloor \frac{vw}{2} \rfloor$. It suffices to prove that the number of subgraphs employed in an $N(v+w, v; 4, 1)$ other than triangles, kites, and 4-cycles is at least $\lceil \binom{v}{2} - \frac{1}{2}vw \rceil = \binom{v}{2} - \lfloor \frac{1}{2}vw \rfloor$. By Lemma 4.1, this is a lower bound on the number of positive edges in any such decomposition; because each positive edge lies in a different subgraph of the decomposition, the lower bound follows.

Now we turn to the upper bounds. For the first statement, because an $\mathscr{ON}(v+w, v; 4, 1)$ is also an $\mathscr{ON}(v+w, v-1; 4, 1)$, it suffices to consider $v \in \{w, w+1\}$. When $v = w$, write $v = 4s+t$ with $t \in \{0, 3, 5, 6\}$. Form on V a complete multipartite graph with s classes of size four and one class of size t. Replace edge $e = \{x, y\}$ of this graph by the 4-cycle (x, y, a_x, a_y). On $\{x_1, \ldots, x_\ell, a_{x_1}, \ldots, a_{x_\ell}\}$ whenever $\{x_1, \ldots, x_\ell\}$ forms a class of the multipartite graph, place a decomposition that is optimal for drop cost and uses 4, 7, 12, and 17 wavelengths when ℓ is 3, 4, 5, or 6, respectively (see Section 4.6.1).

Now let $v = w + 1$. Let $V = \{0, \ldots, v-1\}$ and $W = \{a_0, \ldots, a_{v-2}\}$. Form triangles $(i, i+1, a_i)$ for $0 \le i < v-1$. Then form 4-cycles $(i, j+1, a_i, a_j)$ for $0 \le i < j \le v-2$.

Finally, suppose that $v \ge w + 2$. When v is even, form a 1-factorization $F_0, \ldots, F_{v-2}$ on V. For $0 \le i < w$, let $\{e_{ij} : 1 \le j \le \frac{v}{2}\}$ be the edges of F_i, and form triangles $T_{ij} = \{a_i\} \cup e_{ij}$. Now for $0 \le i < w$; $1 \le j \le \lfloor \frac{w}{2} \rfloor$; and furthermore $j \ne \frac{w}{2}$ if $i \ge \frac{w}{2}$ and w is even, adjoin edge $\{a_i, a_{i+j \bmod w}\}$ to T_{ij} to form a kite. All edges of 1-factors $\{F_i : w \le i < v-1\}$ are taken as K_2s.

When v is odd, form a near 1-factorization $F_0, \ldots, F_{v-1}$ on V, in which F_{v-1} contains the edges $\{\{2h, 2h+1\} : 0 \le h < \frac{v-1}{2}\}$, and near 1-factor F_i misses vertex i for $0 \le i < v$. Then form 4-cycles $(2h, 2h+1, a_{2h+1}, a_{2h})$ for $0 \le h < \lfloor \frac{w}{2} \rfloor$. For $0 \le i < w$, let $\{e_{ij} : 1 \le j \le \frac{v-1}{2}\}$ be the edges of F_i, and form triangles $T_{ij} = \{a_i\} \cup e_{ij}$. Without loss of generality we assume that $w-1 \in e_{01}$; when w is odd, adjoin $\{w-1, a_{w-1}\}$ to T_{01} to form a kite. Now for $0 \le i < w$; $1 \le j \le \lfloor \frac{w}{2} \rfloor$; and furthermore $j \ne \frac{w}{2}$ if $i \ge \frac{w}{2}$ and w is even and $j \ne 1$ if $i = 2h$ for $0 \le h < \lfloor \frac{w}{2} \rfloor$, adjoin edge $\{a_i, a_{i+j \bmod w}\}$ to T_{ij} to form a kite. All edges of near 1-factors $\{F_i : w \le i < v-1\}$ and the $\frac{v-1}{2} - \lfloor \frac{w}{2} \rfloor$ remaining edges of F_{v-1} are taken as K_2s.

When $v \geq w+1$, each subgraph contains exactly one edge on V and so their number is $\binom{v}{2}$. This fact is later used to prove Theorem 4.4. □

4.3.2 $\mathscr{MCN}(n, v; 4, 1)$

Theorem 4.3 *Let $v + w \geq 5$. For $C' = 1$ and $v \leq w$,*

$$\textit{wavecost}\,\mathscr{MCN}(v+w, v; 4, 1) = \textit{wavecost}\,\mathscr{MCN}(v+w, 4).$$

Proof: We need only treat the cases when $v \in \{w, w-1\}$; the case with $v = w$ is handled in the proof of Theorem 4.2. When $v = w - 1$, the argument is identical to that proof, except that we choose $v = 4s + t$ with $t \in \{0, 1, 2, 3\}$ and place decompositions on $\{x_1, \ldots, x_\ell, a_{x_1}, \ldots, a_{x_\ell}, a_v\}$ instead, with 1,3,6,9 wavelengths when $\ell = 1, 2, 3, 4$ respectively (see Section 4.6.2). □

Theorem 4.4 *When $v > w$,*

$$\textit{wavecost}\,\mathscr{MCN}(v+w, v; 4, 1) = \binom{v}{2}.$$

Proof: Since every edge on V appears on a different wavelength, $\binom{v}{2}$ is a lower bound. As noted in the proof of Theorem 4.2 the constructions given there meet this bound. □

The solutions used from Theorem 4.2 are (essentially) the only ones to minimize the number of graphs in an $\mathscr{CN}(v+w, v; 4, 1)$ with $v > w$. However, perhaps surprisingly they are not the only ones to minimize the number of wavelengths. To see this, consider a $\mathscr{CN}(v+w, v; 4, 1)$ with $v > w > 2$ from Theorem 4.2. Remove edges $\{a_0, a_1\}$, $\{a_0, a_2\}$, and $\{a_1, a_2\}$ from their kites, and form a triangle from them. This does not change the drop cost, so the result is also an $\mathscr{CN}(v+w, v; 4, 1)$. It has one more graph than the original. Despite this, it does not need an additional wavelength, since the triangle (a_0, a_1, a_2) can share a wavelength with an edge on V. In this case, while minimizing the number of connected graphs serves to minimize the number of wavelengths, it is not the only way to do so.

4.4 Case $C' = 2$

4.4.1 $\mathscr{CN}(n, v; 4, 2)$

Theorem 4.5 *Let $v + w \geq 5$ and v be even.*

1. *When $v \leq 2w$, cost $\mathscr{CN}(v+w, v; 4, 2) = $ cost $\mathscr{CN}(v+w, 4)$.*
2. *When $v \geq 2w+2$, cost $\mathscr{CN}(v+w, v; 4, 2) = \binom{v+w}{2} + \lceil \frac{1}{2}\binom{v}{2} \rceil - \frac{vw}{2} + \delta$, where $\delta = 1$ if $w = 4$ or if $w = 2$ and $v \equiv 0 \pmod 4$, and $\delta = 0$ otherwise.*

Proof: By Lemma 4.1, $\binom{v}{2} - vw$ is a lower bound on the number of positive edges in any $N(v+w, v; 4, 2)$; every subgraph of the decomposition containing a positive edge contains at most two positive edges. So the number of subgraphs employed in an $N(v+w, v; 4, 2)$ other than triangles, kites, and 4-cycles is at least $\lceil \frac{1}{2}\left(\binom{v}{2} - vw\right)\rceil$. The lower bound follows for $w \neq 2, 4$.

As in the proof of Lemma 4.1, denote by α (resp. β) the number of subgraphs containing 2 (resp 1) neutral edges and so at least two cross edges. We have $2\alpha + \beta \leq 2\alpha + 2\beta \leq vw$. Equality in the lower bound, when $v \equiv 0 \pmod 4$, arises only when $\beta = 0$ and therefore to meet the bound an $\mathscr{ON}(w, 4)$ must be placed on W implying that $\delta = 1$ if $w = 2$ or 4. When $v \equiv 2 \pmod 4$, we can have $2\alpha + \beta = vw - 1$ and so $\beta = 1$. We can use an edge on W in a graph with an edge on V. But when $w = 4$, the five edges that would remain on W require drop cost 6, and so $\delta = 1$.

Now we turn to the upper bounds. If $w \geq v - 1$, apply Theorem 4.2. Suppose that $w \leq v - 2$. Let $V = \{0, \ldots, 2t-1\}$ and $W = \{a_0, \ldots, a_{w-1}\}$. Place an $\mathscr{ON}(w, 4)$ on W. Form a 1-factorization on V containing factors $\{F_0, \ldots, F_{w-1}, G_0, \ldots, G_{2t-2-w}\}$ in which the last two 1-factors are $\{\{2h, 2h+1\} : 0 \leq h < t\}$ and $\{\{2h+1, 2h+2 \bmod 2t\} : 0 \leq h < t\}$, whose union is a Hamilton cycle. For $0 \leq i < w$, form triangles T_{ij} by adding a_i to each edge $e_{ij} \in F_i$. For $0 \leq i < \min(w, 2t-1-w)$, observe that $H_i = F_i \cup G_i$ is a 2-factor containing even cycles. Hence there is a bijection σ mapping edges of F_i to edges of G_i so that e and $\sigma(e)$ share a vertex. Adjoin edge $\sigma(e_{ij})$ to the triangle T_{ij} to form a kite. In this way, all edges between V and W appear in triangles or kites, and all edges on V are employed when $v \leq 2w$. When $v \geq 2w + 2$, the edges remaining on V are those of the factors $G_w, \ldots, G_{v-2-w}$.

When $v \neq 2w + 2$, the union of these edges is connected because the union of the last two is connected, and hence it can be partitioned into P_3s (and one P_2 when $v \equiv 2 \pmod 4$) [66, 207]. When $w = 2$ and $v \equiv 2 \pmod 4$, the drop cost can be reduced by 1 as follows. Let $\{x, y\}$ be the P_2 in the decomposition, and let $\{x, z\} \in G_0$. Let T be the triangle obtained by removing $\{x, z\}$ from its kite. Add $\{a_0, a_1\}$ to T to form a kite. Add the P_3 $[y, x, z]$. In this way two isolated P_2s are replaced by a P_3, lowering the drop cost by 1.

When $v = 2w + 2$, we use a variant of this construction. Let R be a graph with vertex set V that is isomorphic to $\frac{v}{4}$ K_4s when $v \equiv 0 \pmod 4$ and to $\frac{v-6}{4}$ K_4s and one $K_{3,3}$ when $v \equiv 2 \pmod 4$. Let $F_1, \ldots, F_{w-1}, G_1, \ldots, G_{w-1}$ be the 1-factors of a 1-factorization of the complement of R (one always exists [189]). Proceed as above to form kites using a_i for $1 \leq i < w$ and the edges of F_i and G_i. For each K_4 of R with vertices $\{p, q, r, s\}$, form kites $(a_0, q, p; r)$ and $(a_0, r, s; p)$. Then add the P_3 $[r, q, s]$. If R contains a $K_{3,3}$ with bipartition $\{\{p, q, r\}, \{s, t, u\}\}$, add kites $(a_0, s, p; t)$, $(a_0, q, t; r)$, and $(a_0, r, u; p)$. What remains is the P_4 $[r, s; q, u]$, which can be partitioned into a P_2 and a P_3. □

In order to treat the odd case, we establish an easy preliminary result:

Lemma 4.2 *Let $w > 3$ be a positive integer. The graph on w vertices containing all edges except for $\lfloor \frac{w}{2} \rfloor$ disjoint edges (i.e., $K_w \setminus \lfloor \frac{w}{2} \rfloor K_2$) can be partitioned into*

1. *4-cycles when w is even;*
2. *kites and 4-cycles when $w \equiv 1 \pmod 4$; and*
3. *kites, 4-cycles, and exactly two triangles when $w \equiv 3 \pmod 4$.*

Proof: Let $W = \{a_0, \ldots, a_{w-1}\}$. When w is even, form 4-cycles $\{(a_{2i}, a_{2j}, a_{2i+1}, a_{2j+1}) : 0 \le i < j < \frac{w}{2}\}$ leaving uncovered the $\frac{w}{2}$ edges $\{a_{2i}, a_{2i+1}\}$. (This is also a consequence of a much more general result in [129].)

When w is odd, the proof is by induction on w by adding four new vertices. So we provide two base cases for the induction to cover all odd values of w.

For $w = 5$, $K_5 \setminus \{\{a_0, a_1\}, \{a_2, a_3\}\}$ can be partitioned into the two kites $(a_2, a_4, a_0; a_3)$ and $(a_3, a_4, a_1; a_2)$.

For $w = 7$, $K_7 \setminus \{\{a_0, a_1\}, \{a_2, a_3\}, \{a_4, a_5\}\}$ can be partitioned into the kites $(a_3, a_6, a_0; a_5)$, $(a_1, a_6, a_4; a_3)$ and $(a_5, a_6, a_2; a_1)$, and the K_3s (a_0, a_2, a_4) and (a_1, a_3, a_5).

By induction consider an optimal decomposition of $K_w - F$, with $F = \{\{a_{2h}, a_{2h+1}\} : 0 \le h < \frac{w-1}{2}\}$. Add four vertices $a_w, a_{w+1}, a_{w+2}, a_{w+3}$. Add the C_4s $(a_{2h}, a_w, a_{2h+1}, a_{w+1})$ and $(a_{2h}, a_{w+2}, a_{2h+1}, a_{w+3})$ where $0 \le h < \frac{w-1}{2}$. Cover the edges of the K_5 on $\{a_{w-1}, a_w, a_{w+1}, a_{w+2}, a_{w+3}\}$ minus the edges $\{a_{w-1}, a_w\}$ and $\{a_{w+1}, a_{w+2}\}$, using two kites as shown for the case when $w = 5$. □

Theorem 4.6 *Let $v + w \ge 5$ and v be odd.*

1. *When $v \le 2w - 1$, cost $\mathscr{CN}(v + w, v; 4, 2)$ = cost $\mathscr{CN}(v + w, 4)$.*
2. *When $v \ge 2w + 1$, cost $\mathscr{CN}(v + w, v; 4, 2) = \binom{v+w}{2} + \lceil \frac{1}{2} \left(\binom{v}{2} - vw + \lceil \frac{w}{2} \rceil \right) \rceil + \delta$, where $\delta = 1$ if $w = 3$ and $v \equiv 3 \pmod 4$, 0 otherwise.*

Proof: To prove the lower bound, it suffices to prove that the number of subgraphs employed in an $N(v + w, v; 4, 2)$ other than triangles, kites, and 4-cycles is at least $\lceil \frac{1}{2} \left(\binom{v}{2} - vw + \lceil \frac{w}{2} \rceil \right) \rceil$. As in the proof of Theorem 4.5, this follows from Lemma 4.1. When $w = 3$ and $v \equiv 3 \pmod 4$, at least $\binom{v}{2} - 3v + 2$ edges are positive, an even number. To meet the bound, exactly one cross edge remains and exactly two edges on W remain. These necessitate a further graph that is not a triangle, kite, or 4-cycle.

Now we turn to the upper bounds. By Theorem 4.5, cost $\mathscr{CN}((v+1)+(w-1), v+1; 4, 2)$ = cost $\mathscr{CN}(v + w, 4)$ when $v \le 2w - 3$. So suppose that $v \ge 2w - 1$. Write $v = 2t + 1$.

When $w = t + 1$, form a near 1-factorization on V consisting of $2t + 1$ near 1-factors, $F_0, \ldots, F_t, G_0, \ldots, G_{t-1}$. Without loss of generality, F_i misses vertex i for $0 \le i \le t$, and F_t contains the edges $\{\{k, t + k + 1\} : 0 \le k < t\}$. The union of any two near 1-factors contains a nonnegative number of even cycles and a path with an even number of edges. For $0 \le i \le t$, form triangles T_{ij} by adding a_i to each edge $e_{ij} \in F_i$. As in the proof of Theorem 4.5, for $0 \le i < t$, use the edges of G_i to convert every triangle T_{ij} into a kite. Then add edge $\{i, a_i\}$ to triangle T_{ti} constructed from edge $\{i, t + 1 + i\}$. What remains is the single edge $\{t, a_t\}$ together with all edges on W.

When $w \notin \{2, 4\}$, place an $\mathscr{CN}(w, 4)$ on W of cost $\binom{w}{2}$ so that a_t appears in a triangle in the decomposition, and use the edge $\{t, a_t\}$ to convert this to a kite. We use a decomposition having $1 \le \delta \le 4$ triangles, therefore getting a solution with at most 3 triangles. Such a decomposition exists by Theorem 4.1 if $w \not\equiv 0, 1 \pmod 8$. If $w \equiv 0, 1 \pmod 8$ we build a solution using 4 triangles as follows. If $w \equiv 1 \pmod 8$, form an $\mathscr{CN}(w - 2, 4)$ on vertices

$\{0,\ldots,w-3\}$ with 3 triangles. Add the triangle $(w-3,w-2,w-1)$ and the 4-cycles $\{(2h,w-2,2h+1,w-1) : 0 \le h < \frac{w-3}{2}\}$. For $w = 8$ a solution with 4 triangles is given in Section 4.6.3. In general, for $w \equiv 0 \pmod 8$, form an $\mathscr{OMN}(w-8,4)$ on vertices $\{0,\ldots,w-9\}$ with 4 triangles. Add the 4-cycles $\{(2h,w-2j,2h+1,w-2j+1) : 0 \le h < \frac{w-8}{2}\}; 1 \le j \le 4$ and an $\mathscr{OMN}(8,4)$ without triangles on the 8 vertices $\{w-8,\ldots,w-1\}$.

Two values for w remain. When $w = 2$, an $\mathscr{OMN}(5,3;4,1)$ is also an $\mathscr{OMN}(5,3;4,2)$. The case when $v = 7$ and $w = 4$ is given in Section 4.6.3. The solution given has only 1 triangle.

Henceforth $w \le t$. For $t > 2$, form a near 1-factorization $\{F_0,\ldots,F_{w-1},G_0,\ldots,G_{2t-1-w}\}$ of $K_v \setminus C_t$, where C_t is the t-cycle on $(0,1,\ldots,t-1)$; such a factorization exists [179]. Name the factors so that the missing vertex in F_i is $\lfloor i/2 \rfloor$ for $0 \le i < w$ (this can be done, as every vertex i satisfying $0 \le i < t$ is the missing vertex in two of the near 1-factors). Form triangles using $F_0,\ldots,F_{w-1}$ and convert to kites using $G_0,\ldots,G_{w-1}$ as before. There remain $2(t-w)$ near 1-factors $G_w,\ldots,G_{2t-1-w}$. For $0 \le h < t-w$, $G_{w+2h} \cup G_{w+2h+1}$ contains even cycles and an even path, and so partitions into P_3s. Then the edges remaining are (1) the edges of the t-cycle; (2) the edges $\{\{\lfloor i/2 \rfloor, a_i\} : 0 \le i < w\}$; and (3) all edges on W. For $0 \le i < \lfloor \frac{w}{2} \rfloor$, form triangle (i,a_{2i},a_{2i+1}) and add edge $\{i,i+1\}$ to convert it to a kite. Edges $\{\{i,i+1 \bmod t\} : \lfloor \frac{w}{2} \rfloor \le i < t\}$ of the cycle remain from (1); edge $\{\frac{w-1}{2}, a_{w-1}\}$ remains when w is odd, and no edge remains when w is even, from (2); and all edges excepting a set of $\lfloor \frac{w}{2} \rfloor$ disjoint edges on W remain.

When $w \ne 3$, we partition the remaining edges in (1) (which form a path of length $t - \lfloor \frac{w}{2} \rfloor$), into P_3s when $t - \lfloor \frac{w}{2} \rfloor$ is even, and into P_3s and the P_2 $\{0,t-1\}$ when $t - \lfloor \frac{w}{2} \rfloor$ is odd. We adjoin edge $\{\frac{w-1}{2}, a_{w-1}\}$ to the P_3 (from the t-cycle) containing the vertex $\frac{w-1}{2}$ to form a P_4. Finally, we apply Lemma 4.2 to exhaust the remaining edges on W.

When $w = 3$, the remaining edges are those of the path $[0,t-1,t-2,\ldots,2,1,a_2]$ and edges $\{\{a_2,a_0\},\{a_2,a_1\}\}$. Include $\{\{1,2\},\{1,a_2\},\{a_2,a_0\},\{a_2,a_1\}\}$ in the decomposition, and partition the remainder into P_3s and, when $v \equiv 3 \pmod 4$, one P_2 $\{0,t-1\}$.

The case when $t = 2$ is done in Example 4.2 (the construction is exactly that given above, except that we start with a near 1-factorization of $K_5 \setminus \{\{0,1\},\{0,3\}\}$). □

4.4.2 $\mathscr{MOMN}(n,v;4,2)$

Theorem 4.7 *For $C' = 2$ and $v \le 2w$,*

$$\mathit{wavecost}\,\mathscr{MOMN}(v+w,v;4,2) = \mathit{wavecost}\,\mathscr{MOMN}(v+w,4).$$

Proof: It suffices to prove the statement for $v \in \{2w-2,2w-1,2w\}$. When $v = 2w-1$, apply the construction given in the proof of Theorem 4.6, where we noted that there are at most 3 triangles. The proof of Theorem 4.6 provides explicit solutions when $w \in \{2,4\}$.

Now suppose that $v = 2w$. In the proof of Theorem 4.5, $\frac{v}{2} = w$ triangles containing one edge on V and two edges between a vertex of V and a_{w-1} remain. Then convert $w-1$ triangles to kites using edges on W incident to a_{w-1}. That leaves one triangle. When the remaining edges on the $w-1$ vertices of W support a $\mathscr{MOMN}(w-1,4)$ that contains at most two triangles, we are done. It remains to treat the cases when $w-1 \equiv 2,7 \pmod 8$ or

$w-1=4$.For the first case, let x be one vertex of the triangle left containing a_{w-1}, namely (a_{w-1},x,y). Consider the pendant edge $\{x,t\}\in G_{w-2}$ used in a kite containing a_{w-2}. Delete $\{x,t\}$ from this kite and adjoin $\{a_{w-3},a_{w-2}\}$ to the unique triangle so formed forming another kite. Finally adjoin $\{x,t\}$ to the triangle (a_{w-1},x,y). Proceed as before, but partition all edges on $\{a_0,\ldots,a_{w-2}\}$ except edge $\{a_{w-3},a_{w-2}\}$ into 4-cycles and kites. The case when $w-1=4$ is similar, but we leave three of the triangles arising from F_{w-1} and partition $K_5\setminus P_3$ into two kites.

Now suppose that $v=2w-2$. We do a construction similar to that above. In the proof of Theorem 4.5, there remain $3\frac{v}{2}=3(w-1)$ triangles joining a_{w-3} (resp. a_{w-2},a_{w-1}) to F_{w-3} (resp. F_{w-2},F_{w-1}). Then convert the $w-1$ triangles containing a_{w-1} to kites using edges on W incident to a_{w-1}, $w-2$ triangles containing a_{w-2} to kites using the remaining edges on W incident to a_{w-2}, and $w-3$ triangles containing a_{w-3} to kites using edges on W incident to a_{w-3}. That leaves three triangles. So, if $w-3\equiv 0,1 \pmod 8$ we are done. Otherwise, as above, choose in each of the three remaining triangles vertices x_1,x_2,x_3; consider the edges $\{x_1,t_1\}$ (resp. $\{x_2,t_2\}$) appearing in the kites containing a_{w-4} and x_1 (resp. a_{w-4} and x_2), and the edge $\{x_3,t_3\}$ in the kite containing a_{w-5} and x_3. Delete these edges and adjoin them to the three remaining triangles. Finally adjoin the edges $\{a_{w-4},a_{w-5}\}$ and $\{a_{w-4},a_{w-6}\}$ to the two triangles obtained from the two kites containing a_{w-4}, and adjoin the edge $\{a_{w-5},a_{w-6}\}$ to the triangle obtained from the kite containing a_{w-5}. Proceed as before, but partition all edges on $\{a_0,\ldots,a_{w-4}\}$ except the triangle $(a_{w-6},a_{w-5},a_{w-4})$ into 4-cycles and kites. □

Theorem 4.8 *1. When $v>2w$ is even,*

$$wavecost\,\mathscr{MCN}(v+w,v;4,2)=\left\lceil\left(2\binom{v}{2}+\binom{w}{2}\right)/4\right\rceil.$$

2. When $v>2w$ is odd,

$$wavecost\,\mathscr{MCN}(v+w,v;4,2)=\left\lceil\left(2\binom{v}{2}+\frac{(w-1)(w+1)}{2}\right)/4\right\rceil.$$

Proof: First we treat the case when v is even. Then (by Theorem 4.5) an $\mathscr{CN}(v+w,v;4,2)$ must employ vw or $vw-1$ neutral edges, using all vw edges between V and W. Each such graph uses two edges on V and none on W, except that a single graph may use one on V and one on W. Now the edges of V must appear on $\lceil\frac{1}{2}\binom{v}{2}\rceil$ different wavelengths, and these wavelengths use at most one edge on W (when $v\equiv 2 \pmod 4$). Thus at least $\lceil\binom{w}{2}/4\rceil$ additional wavelengths are needed when $v\equiv 0 \pmod 4$, for a total of $\lceil\binom{v}{2}/2+\binom{w}{2}/4\rceil$. When $v\equiv 2 \pmod 4$, at least $\lceil(\binom{w}{2}-1)/4\rceil$ additional wavelengths are needed; again the total is $\lceil\binom{v}{2}/2+\binom{w}{2}/4\rceil$. Theorem 4.5 realizes this bound.

When v is odd, first suppose that w is even. In order to realize the bound of Theorem 4.6 for drop cost, by Lemma 4.1, $\frac{w}{2}$ neutral edges appear in subgraphs with one neutral edge and all other neutral edges appear in subgraphs with two. In both cases, two edges between V and W are consumed by such a subgraph. When two neutral edges are used, no edge on W can be used ; when one neutral edge is used, one edge on W can also be

used. It follows that the number of wavelengths is at least $\frac{1}{2}(\binom{v}{2} - \frac{w}{2}) + \frac{w}{2} + \frac{1}{4}(\binom{w}{2} - \frac{w}{2})$. This establishes the lower bound. The case when w is odd is similar. The proof of Theorem 4.6 gives constructions with at most 3 triangles and so establishes the upper bound except when $v \equiv 1 \pmod 4$ and $w \equiv 3 \pmod 4$, $w \neq 3$, where the construction employs one more graph than the number of wavelengths permitted. However, one graph included is the P_2 $\{0, t-1\}$, and in the decomposition on W, there is a triangle. These can be placed on the same wavelength to realize the bound. □

When $v \equiv 1 \pmod 4$ and $w \equiv 3 \pmod 4$, $w \neq 3$, we place a disconnected graph, $P_2 \cup K_3$, on one wavelength in order to meet the bound. The construction of Theorem 4.6 could be modified to avoid this by instead using a decomposition of $K_w \setminus (K_3 \cup \frac{w-3}{2}K_2)$ into 4-cycles and kites, and using the strategy used in the case for $w = 3$. In this way, one could prove the slightly stronger result that the number of (connected) subgraphs in the decomposition matches the lower bound on number of wavelengths needed.

In Theorem 4.4, the number of wavelengths and the drop cost are minimized simultaneously by the constructions given; each constructed $\mathscr{OMN}(v+w, v; 4, 1)$ has not only the minimum drop cost but also the minimum number of wavelengths over all $N(v+w, v; 4, 1)$s. This is not the case in Theorem 4.8. For example, when $v > (1 + \sqrt{2})w$, it is easy to construct an $N(v + w, v; 4, 2)$ that employs only $\lceil \binom{v}{2}/2 \rceil$ wavelengths, which is often much less than are used in Theorem 4.8. We emphasize therefore that a $\mathscr{MOMN}(v + w, v; 4, 2)$ minimizes the number of wavelengths over all $\mathscr{OMN}(v + w, v; 4, 2)$s, *not necessarily* over all $N(v + w, v; 4, 2)$s.

4.5 Case $C' = 3$

4.5.1 $\mathscr{OMN}(n, v; 4, 3)$

Theorem 4.9 *Let $v + w \geq 5$.*

1. *When $w \geq 1$, cost $\mathscr{OMN}(v + w, v; 4, 3)$ = cost $\mathscr{OMN}(v + w, 4)$.*
2. *cost $\mathscr{OMN}(v + 0, v; 4, 3)$ = cost $\mathscr{OMN}(v, 3)$.*

Proof: The second statement is trivial. Moreover *cost* $\mathscr{OMN}(n, 4)$ = *cost* $\mathscr{OMN}(n, 3)$ when $n \equiv 1, 3 \pmod 6$, and hence the first statement holds when $v + w \equiv 1, 3 \pmod 6$. To complete the proof it suffices to treat the upper bound when $w = 1$.

When $v + 1 \equiv 5 \pmod 6$, there is a maximal partial triple system $(X, \mathcal{B})$ with $|X| = v + 1$ covering all edges except those in the 4-cycle (r, x, y, z). Set $W = \{r\}$, $V = X \setminus W$, and add the 4-cycle to the decomposition to obtain an $\mathscr{OMN}(v + 1, v; 4, 3)$.

When $v \equiv 1, 5 \pmod 6$, set $\ell = v - 1$ and when $v \equiv 3 \pmod 6$ set $\ell = v - 3$. Then ℓ is even. Form a maximal partial triple system $(V, \mathcal{B})$, $|V| = v$, covering all edges except those in an ℓ-cycle $(0, 1, \ldots, \ell - 1)$ [82]. Add a vertex a_0 and form kites $(a_0, 2i, 2i + 1; (2i + 2) \bmod \ell)$ for $0 \leq i < \frac{\ell}{2}$. For $i \in \{\ell, \ldots, v - 1\}$, choose a triple $B_i \in \mathcal{B}$ so that $i \in B_i$ and $B_i = B_j$ only if $i = j$. Add $\{a_0, i\}$ to B_i to form a kite. This yields an $\mathscr{OMN}(v + 1, v; 4, 3)$. □

4.5.2 $\mathscr{MON}(n, v; 4, 3)$

We focus first on lower bounds in Section 4.5.2 and then we provide constructions attaining these lower bounds in Section 4.5.2.

Lower Bounds

When $C' = 3$, Theorem 4.9 makes no attempt to minimize the number of wavelengths. We focus on this case here. Except when $n \in \{2, 4\}$ or $v = n$, $cost\,\mathscr{ON}(n, v; 4, 3) = \binom{n}{2}$, and every graph in an $\mathscr{ON}(n, v; 4, 3)$ is a triangle, kite, or 4-cycle. Let δ, κ, and γ denote the numbers of triangles, kites, and 4-cycles in the grooming, respectively. Then $3\delta + 4\kappa + 4\gamma = \binom{n}{2}$, and the number of wavelengths is $\delta + \kappa + \gamma$. Thus in order to minimize the number of wavelengths, we must minimize the number δ of triangles. We focus on this equivalent problem henceforth.

In an $\mathscr{ON}(n, v; 4, 3)$, for $0 \le i \le 3$ and $0 \le j \le 4$, let δ_{ij}, κ_{ij}, and γ_{ij} denote the number of triangles, kites, and 4-cycles, respectively, each having i edges on V and j edges between V and W. The only counts that can be nonzero are δ_{00}, δ_{02}, δ_{12}, δ_{30}; κ_{00}, κ_{01}, κ_{02}, κ_{03}, κ_{12}, κ_{13}, κ_{22}, κ_{31}; γ_{00}, γ_{02}, γ_{04}, γ_{12}, γ_{22}. We write $\sigma_{ij} = \kappa_{ij} + \gamma_{ij}$ when we do not need to distinguish kites and 4-cycles. Our objective is to minimize $\delta_{00} + \delta_{02} + \delta_{12} + \delta_{30}$ subject to certain constraints; we adopt the strategy of [81] and treat this as a linear program.

Let $\varepsilon = 0$ when $v \equiv 1, 3 \pmod 6$, $\varepsilon = 2$ when $v \equiv 5 \pmod 6$, and $\varepsilon = \frac{v}{2}$ when $v \equiv 0 \pmod 2$. We specify the linear program in Figure 4.1. The first row lists the primal variables. The second lists coefficients of the objective function to be minimized. The remainder list the coefficients of linear inequalities, with the final column providing the *lower bound* on the linear combination specified. The first inequality states that the number of edges on V used is at least the total number on V, while the second specifies that the number of edges used between V and W is at most the total number between V and W. For the third, when $v \equiv 5 \pmod 6$ at least four edges on V are not in triangles, and so at least two graphs containing edges of V do not have a triangle on V; when $v \equiv 0 \pmod 2$ every graph can induce at most two odd degree vertices on V, yet all are odd in the decomposition.

δ_{30}	δ_{12}	δ_{02}	δ_{00}	κ_{31}	σ_{22}	κ_{13}	σ_{12}	γ_{04}	κ_{03}	σ_{02}	κ_{01}	σ_{00}	
1	1	1	1	0	0	0	0	0	0	0	0	0	
3	1	0	0	3	2	1	1	0	0	0	0	0	$\binom{v}{2}$
0	-2	-2	0	-1	-2	-3	-2	-4	-3	-2	-1	0	$-vw$
0	1	0	0	0	1	1	1	0	0	0	0	0	ε

Figure 4.1: The linear program for $\mathscr{ON}(n, v; 4, 3)$.

We do not solve this linear program. Rather we derive lower bounds by considering its dual. Let y_1, y_2, and y_3 be the dual variables. A dual feasible solution has $y_1 = \frac{1}{3}$, $y_2 = 1$, and $y_3 = \frac{4}{3}$, yielding a dual objective function value of $\frac{1}{6}v(v-1) - vw + \frac{4}{3}\varepsilon$. Recall that every dual feasible solution gives a lower bound on all primal feasible solutions

On the other hand, $3\delta \equiv \binom{n}{2} \pmod 4$ and so $\delta \equiv 9\delta \equiv 3\binom{n}{2} \pmod 4$. The value of $3\binom{n}{2} \pmod 4$ is in fact the value of t given in Theorem 4.1. Therefore if x is a lower bound on δ in an $\mathscr{ONN}(n, v; 4, 3)$, so is $\langle x \rangle_n$, where $\langle x \rangle_n$ denotes the smallest nonnegative integer $\overline{x}$ such that $\overline{x} \geq x$ and $\overline{x} \equiv 3\binom{n}{2} \pmod 4$.

The discussion above proves the general lower bound on the number of triangles:

Theorem 4.10 *Let $v + w \geq 5$, and let*

$$L(v,w) = \begin{cases} \frac{1}{6}v(v-1) - vw & \text{if } v \equiv 1,3 \pmod 6 \\ \frac{1}{6}v(v-1) - vw + \frac{8}{3} & \text{if } v \equiv 5 \pmod 6 \\ \frac{1}{6}v(v+3) - vw & \text{if } v \equiv 0 \pmod 2 \end{cases}$$

Then the number of triangles in an $\mathscr{ONN}(v + w, v; 4, 3)$ is at least

$$\delta_{\min}(v,w) = \langle L(v,w) \rangle_{v+w}$$

Remark 4.1 *In particular, if v is odd and $w \geq \lceil \frac{v-1}{6} \rceil$ or if v is even and $w \geq \lceil \frac{v-4}{6} \rceil$, then $L(v,w) \leq 0$ and the minimum number of triangles is $\delta_{\min}(v,w) = \langle 0 \rangle_{v+w} \leq 3$.*

Upper Bounds

We first state two simple lemmas to be used intensively in the proof of Theorem 4.11. The following result shows that in fact we do not need to check *exactly* that the number of triangles of an optimal construction meets the bound of Theorem 4.10.

Lemma 4.3 *Any $\mathscr{ONN}(v + w, v; 4, 3)$ is a $\mathscr{MONN}(v + w, v; 4, 3)$ if the number of triangles that it contains is at most* $\max(3, \lceil L(v,w) \rceil + 3)$.

Proof: In the closed interval $[\lceil L(v,w) \rceil, \lceil L(v,w) \rceil + 3]$ there is exactly one integer congruent to $3\binom{n}{2} \pmod 4$, and so necessarily exactly one integer equal to $\delta_{\min}(v,w)$. □

Combining Remark 4.1 and Lemma 4.3 we deduce that when v is odd and $w \geq \lceil \frac{v-1}{6} \rceil$ or if v is even and $w \geq \lceil \frac{v-4}{6} \rceil$, to prove the optimality of a construction it is enough to check that there are at most three triangles.

As a prelude to the constructions, let $(V, \mathcal{B})$ be a partial triple system, $V = \{0, \ldots, v-1\}$, and $\mathcal{B} = \{B_1, \ldots, B_b\}$. Let r_i be the number of blocks of $\mathcal{B}$ that contain $i \in V$. A *headset* is a multiset $S = \{s_1, \ldots, s_b\}$ so that $s_k \in B_k$ for $1 \leq k \leq b$, and for $0 \leq i \leq v-1$ the number of occurrences of i in S is $\lfloor \frac{r_i}{3} \rfloor$ or $\lceil \frac{r_i}{3} \rceil$.

Lemma 4.4 *Every partial triple system has a headset.*

Proof: Form a bipartite graph Γ with vertex set $V \cup \mathcal{B}$, and an edge $\{v, B\}$ for $v \in V$ and $B \in \mathcal{B}$ if and only if $v \in B$. The graph Γ admits an equitable 3-edge-colouring [91]; that is, the edges can be coloured green, white, and red so that every vertex of degree d is incident with either $\lfloor d/3 \rfloor$ or $\lceil d/3 \rceil$ edges of each colour. Then for $1 \leq k \leq b$, B_k is incident to exactly three edges, and hence to exactly one edge $\{i_k, B_k\}$ that is green; set $s_k = i_k$. Then $(s_1, \ldots, s_b)$ forms the headset. □

Theorem 4.11 *Let $v + w \geq 5$. When $w \geq 1$,*

$$wavecost\,\mathscr{MCN}(v+w, v; 4, 3) = \left\lceil \left(\binom{v+w}{2} + \delta_{\min}(v, w) \right) / 4 \right\rceil.$$

Proof: The lower bound follows from Theorem 4.10, so we focus on the upper bound.

When $w \geq 1$, an $\mathscr{CN}(v+w, v; 4, 3)$ of cost $\binom{v+w}{2}$ is an $\mathscr{CN}(v+w, v-1; 4, 3)$. Let us show that it suffices to prove the statement for $w \leq \frac{v+9}{6}$ when v is odd, and for $w \leq \frac{v+4}{6}$ when v is even. Equivalently, we show that if it is true for these values of w, then it follows for any w. Note that $\delta_{\min}(v, w) \leq 3$ if $\delta_{\min}(v+1, w-1) \leq 3$.

Indeed, let v be even. If $w = \lfloor \frac{v+4}{6} \rfloor + 1$, the result follows from the case for $v+1$ (odd) and $w - 1 = \lfloor \frac{v+4}{6} \rfloor \leq \frac{v+1+9}{6}$, in which case $\delta_{\min}(v+1, w-1) = \langle 0 \rangle_{v+w}$. If $w = \lfloor \frac{v+4}{6} \rfloor + 2$ it follows from the case for $v+1$ (odd) and $w - 1 = \lfloor \frac{v+4}{6} \rfloor + 1 \leq \frac{v+1+9}{6}$, and $\delta_{\min}(v+1, w-1) = \langle 0 \rangle_{v+w}$. If $w \geq \lfloor \frac{v+4}{6} \rfloor + 3$ it follows from the case for $v+2$ (even) and $w-2$.

Let v be odd. If $w = \lfloor \frac{v+9}{6} \rfloor + 1$ it follows from the case for $v+1$ (even) and $w-1$, which has been already proved (in this case also $\delta_{\min}(v+1, w-1) = \langle 0 \rangle_{v+w}$). If $w \geq \lfloor \frac{v+9}{6} \rfloor + 2$ it follows from the case for $v+2$ (odd) and $w-2$.

In each case, we use the same general prescription. Given a partial triple system $(V, \mathcal{B})$, a headset $S = \{s_1, \ldots, s_b\}$ is formed using Lemma 4.4. Add vertices $W = \{a_0, \ldots, a_{w-1}\}$, a set disjoint from V of size $w \geq 1$. For each i let D_i be a subset of $\{0, \ldots, w-1\}$, which is specified for each subcase, and that satisfies the following property: $|D_i|$ is at most the number of occurrences of i in the headset S. Among the blocks B_k such that $s_k = i$, we choose $|D_i|$ of them, namely the subset $\{B_k^j : j \in D_i\}$, and form $|D_i|$ kites by adding for each $j \in D_i$ the edge $\{a_j, i\}$ to the block B_k^j.

The idea behind the construction is that if we can choose $|D_i| = w$, we use all the edges between V and W leaving a minimum number of triangles in the partition of V (see Case **O1a**). Unfortunately it is not always possible to choose $|D_i| = w$, in particular when w is greater than the number of occurrences of i in the headset. So we distinguish different cases:

Case O1a. $v = 6t+1$ **or** $6t+3$ **and** $w \leq \frac{v-1}{6}$**.** Let $(V, \mathcal{B})$ be a Steiner triple system. For $0 \leq i < v$, let $D_i = \{0, \ldots, w-1\}$. Apply the general prescription. If $v = 6t+1$, i appears t times in S and $w \leq \frac{v-1}{6} = t$. If $v = 6t+3$, i appears t or $t+1$ times in S and $w \leq t$. In both cases $|D_i|$ is at most the number of occurrences of i in S, so the construction applies and all the edges between V and W are used in the kites. All the edges on V are used and $\frac{v(v-1)}{6} - vw$ triangles remain. Finally, it remains to partition the edges of W. When $w \notin \{2, 4\}$, form a $\mathscr{MCN}(w, 4)$ on W, and doing so we have at most $\delta_{\min}$ triangles. If $w = 2$ or $w = 4$ remove edges $\{a_0, 0\}$ and $\{a_1, 0\}$ from their kites and partition K_W together with these edges into a triangle ($w = 2$) or two kites ($w = 4$).

Case O1b. $v = 6t+5$ **and** $w \leq \frac{v-1}{6}$**.** Form a partial triple system $(V, \mathcal{B})$ covering all edges except those in the C_4 $(0, 1, 2, 3)$. For $0 \leq i \leq 3$, let $D_i = \{0, \ldots, w-2\}$ and for $4 \leq i < v$ $D_i = \{0, \ldots, w-1\}$. Apply the general prescription. Add the kites $(a_{w-1}, 1, 2; 3)$

and $(a_{w-1}, 3, 0; 1)$. Here again i appears at least t times in S and $w \le t$. So D_i is at most the number of occurrences of i in S. Again we have used all the edges on V and all the edges between V and W. It remains to partition the edges of W, and this can be done as in the Case **O1a**.

Case O2. $v = 6t + 3$ **and** $w = t + 1$, $v > 3$. Form a partial triple system covering all edges except those on the v-cycle $\{\{i, (i+1) \bmod v\} : 0 \le i < v\}$ [82]. Set $D_i = \{1, \ldots, w-1\}$ for all i. Apply the general prescription. Adjoin edges from a_0 to a partition of the cycle, minus edge $\{0, v-1\}$, into P_3s. The only edge between V and W that remains is $\{a_0, v-1\}$. When an $\mathscr{OMN}(w, 4)$ exists having 1, 2, 3, or 4 triangles, this edge is used to convert a triangle to a kite. This handles all cases except when $w \in \{2, 4\}$. In these cases, remove the pendant edge $\{a_1, v-1\}$ from its kite. When $w = 2$, $\{a_0, a_1, v-1\}$ forms a triangle. When $w = 4$, partition the edges on W together with $\{a_0, v-1\}$ and $\{a_1, v-1\}$ into two kites.

Case O3. $v = 6t + 1$ **and** $w = t + 1$.

When $t = 1$, a $\mathscr{MOMN}(7+2, 7; 4, 3)$ has $\mathcal{B} = \{(0, a_1, a_0; 6), (2, 0, 6; a_1), (3, 0, 4; a_1), (1, 0, 5; a_1), (3, 6, 5; a_0), (4, 6, 1; a_1), (3, 2, 1; a_0), (5, 2, 4; a_0), (a_0, 2, a_1, 3)\}$.

A solution with $t = 2$ is given in Section 4.6.4.

When $t \ge 3$, form a 3-GDD of type 6^t with groups $\{\{6p + q : 0 \le q < 6\} : 0 \le p < t\}$. Let $D_{6p+q} = \{0, \ldots, w-2\} \setminus \{p\}$ for $0 \le p < t$ and $0 \le q < 6$. Apply the general prescription. For $0 \le p < t$, on $\{6p + q : 0 \le q < 6\} \cup \{v-1\} \cup \{a_{w-1}, a_p\}$ place a $\mathscr{MOMN}(7+2, 7; 4, 3)$ obtained from the solution $\mathcal{B}$ for $t = 1$, by replacing q by $6p + q : 0 \le q < 6$, 6 by $v - 1$ a_0 by a_{w-1} and a_1 by a_p; then omit the kite $(a_p, 6p, a_{w-1}; v-1)$. All edges on W remain; the edges $\{a_{w-1}, 6p\}$ and $\{a_p, 6p\}$ remain for $0 \le p < t$, and the edge $\{a_{w-1}, v-1\}$ remains.

Add the kites $(a_{w-2}, 6(w-2), a_{w-1}; v-1)$ and for $0 \le j < w-2 = t-1$ $(6j, a_{w-1}, a_j; a_{w-2})$. If $w - 2 \notin \{2, 4\}$, that is $t \notin \{3, 5\}$, place a $\mathscr{MOMN}(w-2, 4)$ on $W - a_{w-2} - a_{w-1}$. Note that, as $3\binom{w-2}{2} \equiv 3\binom{v+w}{2} \pmod 4$, we have the right number of triangles (at most 3). If $w - 2 \in \{2, 4\}$ remove edges $\{a_0, w-2\}$ and $\{a_1, w-2\}$ from their kites, and partition K_w together with these edges.

Case O4. $v = 6t + 5$ **and** $w = t + 1$.

For $t = 0$, a $\mathscr{MOMN}(5+1, 5; 4, 3)$ has kites $(3, a_0, 0; 1)$, $(1, a_0, 2; 3)$, $(1, 3, 4; a_0)$, and triangle $(0, 2, 4)$.

For $t = 1$, let $V = \{0, \ldots, 10\}$ and $W = \{a_0, a_1\}$. A $\mathscr{MOMN}(11+2, 11; 4, 3)$ is formed by using an $\mathscr{MOMN}(5+1, 5; 4, 3)$ on $\{0, 1, 2, 3, 4\} \cup \{a_0\}$, and a partition of the remaining edges, denoted by $\mathcal{Q}$, into 15 kites and a triangle. So we have two triangles, attaining $\delta_{min}(11, 2)$ as $13 \equiv 5 \pmod 8$. The partition of $\mathcal{Q}$ is as follows: the triangle $(a_0, a_1, 10)$ and the kites $(0, 6, 5; a_0)$, $(1, 8, 6; a_0)$, $(2, 9, 7; a_0)$, $(3, 10, 8; a_0)$, $(4, 6, 9; a_0)$, $(8, 9, 0; a_1)$, $(5, 7, 1; a_1)$, $(5, 8, 2; a_1)$, $(6, 7, 3; a_1)$, $(5, 10, 4; a_1)$, $(3, 9, 5; a_1)$, $(2, 10, 6; a_1)$, $(0, 10, 7; a_1)$, $(4, 7, 8; a_1)$, and $(1, 10, 9; a_1)$.

For $t = 2$, a $\mathscr{MOMN}(17+3, 17; 4, 3)$ is given in Section 4.6.4.

For $t \geq 3$, form a 3-GDD of type 6^t with groups $\{\{6p+q : 0 \leq q < 6\} : 0 \leq p < t\}$. Let $D_{6p+q} = \{0, \ldots, w-2\} \setminus \{p\}$ for $0 \leq p < t$ and $0 \leq q < 6$. Apply the general prescription. There remain uncovered for each p the edges of the set $\mathcal{Q}_p$ obtained from the complete graph on the set of vertices $\{6p+q : 0 \leq q < 6\} \cup \{v-5, v-4, v-3, v-2, v-1\} \cup \{a_{w-1}, a_p\}$ minus the complete graph on $\{v-5, v-4, v-3, v-2, v-1\} \cup \{a_{w-1}\}$.

To deal with the edges of $\mathcal{Q}_p$, we start from a partition of $\mathcal{Q}$, where we replace pendant edges in kites as follows: replace $\{a_1, 4\}$ by $\{a_1, 10\}$, $\{a_0, 8\}$ by $\{a_0, 10\}$, and $\{a_1, 2\}$ by $\{a_0, 8\}$. We delete the triangle $(a_0, a_1, 10)$, resulting in a new partition of $\mathcal{Q}$ into 15 kites and the 3 edges $\{a_0, a_1\}$, $\{a_1, 2\}$, and $\{a_1, 4\}$. Then we obtain a partition of $\mathcal{Q}_p$ by replacing $\{0, 1, 2, 3, 4\}$ by $\{v-5, v-4, v-3, v-2, v-1\}$, $q+5$ by $6p+q$ for $0 \leq q < 6$, a_0 by a_{w-1}, and a_1 by a_p. At the end we get a partition of $\mathcal{Q}_p$ into 15 kites plus the 3 edges $\{a_{w-1}, a_p\}$, $\{a_p, v-3\}$, and $\{a_p, v-1\}$.

Now the $3t$ edges $\{\{a_{w-1}, a_p\}, \{a_p, v-3\}, \{a_p, v-1\} : 0 \leq p < t\}$ plus the uncovered edges of K_W form a K_{t+3} missing a triangle on $\{a_{w-1}, v-3, v-1\}$. If $t+3 \equiv 2, 3, 4, 5, 6, 7 \pmod 8$, use Theorem 4.1 to form a $\mathscr{ON}(t+3, 4)$ having a triangle $(v-3, v-1, a_{w-1})$ and 0, 1, or 2 other triangles; remove the triangle $(v-3, v-1, a_{w-1})$ to complete the solution with 1, 2, or 3 triangles (the triangle $(v-5, v-3, v-1)$ is still present). A variant is needed when $t+3 \equiv 0, 1 \pmod 8$. In these cases, form a $\mathscr{ON}(t+3, 4)$ (having no triangles) in which $(v-3, a_{w-1}, v-1; a_1)$ is a kite. Remove all edges of this kite, and use edge $\{a_1, v-1\}$ to convert triangle $(v-5, v-3, v-1)$ to a kite.

Finally, place a $\mathscr{MON}(5+1, 5; 4, 3)$ on $\{v-5, v-4, v-3, v-2, v-1\} \cup \{a_0\}$. Altogether we have a partition of all the edges using at most 3 triangles.

Case O5. $v = 6t+5$ **and** $w = t+2$**.**

When $t = 0$, partition all edges on $\{0, 1, 2, 3, 4\} \cup \{a_0, a_1\}$ except $\{a_0, a_1\}$ into kites $(3, 1, a_0; 0)$, $(3, 2, a_1; 0)$, $(a_1, 1, 4; 2)$, $(0, 1, 2; a_0)$, and $(3, 0, 4; a_0)$. Then a $\mathscr{MON}(5+2, 5; 4, 3)$ is obtained by removing pendant edges $\{a_0, 0\}$ and $\{a_1, 0\}$ and adding triangle $(a_0, a_1, 0)$.

When $t = 1$, a $\mathscr{MON}(11+3, 11; 4, 3)$ on $\{0, \ldots, 10\} \cup \{a_0, a_1, a_2\}$ is obtained by taking the above partition on $\{0, 1, 2, 3, 4\} \cup \{a_0, a_1\}$, the triangle (a_0, a_1, a_2), and a partition of the remaining edges (which form a graph called $\mathcal{Q}$) into 11 kites and 6 4-cycles as follows: kites $(2, 9, 7; a_0)$, $(4, 5, 10; a_0)$, $(2, 10, 6; a_1)$, $(4, 6, 9; a_2)$, $(7, 10, 0; a_2)$, $(6, 8, 1; a_2)$, $(5, 8, 2; a_2)$, $(5, 9, 3; a_2)$, $(7, 8, 4; a_2)$, $(6, 7, 5; a_2)$, and $(9, 10, 8; a_1)$; and 4-cycles $(0, 6, a_0, 5)$, $(0, 8, a_0, 9)$, $(1, 5, a_1, 7)$, $(1, 9, a_1, 10)$, $(3, 6, a_2, 7)$, and $(3, 8, a_2, 10)$.

A solution with $t = 2$ is given in Section 4.6.4.

When $t \geq 3$, form a 3-GDD of type 6^t with groups $\{\{6p+q : 0 \leq q < 6\} : 0 \leq p < t\}$. Let $D_{6p+q} = \{0, \ldots, w-3\} \setminus \{p\}$ for $0 \leq p < t$ and $0 \leq q < 6$. Apply the general prescription. Add a partition of the complete graph on $\{v-5, v-4, v-3, v-2, v-1\} \cup \{a_{w-2}, a_{w-1}\}$ as in the case when $t = 0$. It remains to partition, for each p, $0 \leq p < t$, the graph $\mathcal{Q}_p$ is obtained from the complete graph on $\{6p+q : 0 \leq q < 6\} \cup \{v-5, v-4, v-3, v-2, v-1\} \cup \{a_{w-2}, a_{w-1}, a_p\}$ minus the complete graph on $\{v-5, v-4, v-3, v-2, v-1\} \cup \{a_{w-2}, a_{w-1}\}$. This partition is obtained from that of $\mathcal{Q}$ by replacing $\{0, 1, 2, 3, 4\}$ by $\{v-5, v-4, v-3, v-2, v-1\}$, a_0 by a_{w-2}, a_1 by a_{w-1}, and a_2 by a_p. What remains is precisely the edges on W, so place a

$\mathcal{MON}(w,4)$ on W to complete the construction.

Case O6. $v = 6t+3$ **and** $w = t+2$**.**

When $t = 0$, a $\mathcal{MON}(3+2,3;4,3)$ has triangles $(a_0,0,1)$ and $\{a_1,1,2\}$ and 4-cycle $(0,2,a_0,a_1)$.

When $t = 1$, on $\{0,\ldots,8\} \cup \{a_0,a_1,a_2\}$, place kites $(2,6,4;a_0)$, $(0,8,4;a_1)$, $(0,5,7;a_1)$, $(3,6,0;a_2)$, $(1,7,4;a_2)$, $(5,8,2;a_2)$, $(1,6,5;a_2)$, $(2,7,3;a_2)$, $(3,8,1;a_2)$, $(3,5,a_0;a_2)$, $(7,a_0,6;a_2)$, $(6,8,a_1;a_2)$, $(7,a_2,8;a_0)$, and 4-cycle $(3,4,5,a_1)$. Adding the blocks of a $\mathcal{MON}(3+2,3;4,3)$ forms a $\mathcal{MON}(9+3,9;4,3)$.

A solution with $t = 2$ is given in Section 4.6.4.

When $t \geq 3$, form a 3-GDD of type 6^t with groups $\{\{6p+j : 0 \leq q < 6\} : 0 \leq p < t\}$. Let $D_{6p+q} = \{0,\ldots,w-3\} \setminus \{p\}$ for $0 \leq p < t$ and $0 \leq q < 6$. Apply the general prescription. For $0 \leq p < t$, on $\{6p+q : 0 \leq q < 6\} \cup \{v-3,v-2,v-1\} \cup \{a_{w-2},a_{w-1},a_p\}$ place a $\mathcal{MON}(9+3,9;4,3)$, omitting a $\mathcal{MON}(3+2,2;4,3)$ on $\{a_{w-2},a_{w-1},v-3,v-2,v-1\}$. Place a $\mathcal{MON}(3+2,2;4,3)$ on $\{a_{w-2},a_{w-1},v-3,v-2,v-1\}$. Remove edges $\{a_0,a_{w-2}\}$ and $\{a_1,a_{w-1}\}$ from their kites, and convert the two triangles in the $\mathcal{MON}(3+2,2;4,3)$ to kites using these. What remains is all edges on $\{a_0,\ldots,a_{w-3}\}$ and everything is in kites or 4-cycles excepting one triangle involving a_0 and one involving a_1. If $w-2 \equiv 0,1,3,6 \pmod 8$, place a $\mathcal{MON}(w-2,4)$ on $\{a_0,\ldots,a_{w-3}\}$. Otherwise partition all edges on $\{a_0,\ldots,a_{w-3}\}$ except $\{a_0,a_2\}$ and $\{a_1,a_2\}$ into kites, 4-cycles, and at most one triangle, and use the last two edges to form kites with the excess triangles involving a_0 and a_1. The partition needed is easily produced for $w-2 \in \{4,5,7,9\}$ and hence by induction for all the required orders.

Case E1. $v \equiv 0 \pmod 2$ **and** $w \leq \frac{v+2}{6}$**.** Write $v = 6t+s$ for $s \in \{0,2,4\}$. Let $L = (V,E)$ be a graph with edges

$$\{\{3i,3i+1\},\{3i,3i+2\},\{3i+1,3i+2\} : 0 \leq i < t\} \cup \{\{i,3t+i\} : 0 \leq i < 3t\},$$

together with $\{6t,6t+1\}$ when $s = 2$ and with $\{\{6t,6t+1\},\{6t,6t+2\},\{6t,6t+3\}\}$ when $s = 4$. Let $(V,\mathcal{B})$ be a partial triple system covering all edges except those in L (this is easily produced). Let $D_i = \{0,\ldots,w-2\}$ for $0 \leq i < v$. Apply the general prescription. For $0 \leq i < t$ and $j \in \{0,1,2\}$, form the 4-cycle $(a_{w-1},3i+((j+1) \bmod 3),3i+j,3t+3i+j)$. When $s = 4$, form 4-cycle $(a_{w-1},6t+2,6t,6t+3)$. When $s \in \{2,4\}$, form a triangle $(a_{w-1},6t,6t+1)$. All edges on V are used and all edges on W remain. All edges between V and W are used. Except when $w \in \{2,4\}$, or $w \equiv 2,7 \pmod 8$ and $v \equiv 2,4 \pmod 6$ form a $\mathcal{MON}(w,4)$ on W to complete the proof. When $w \equiv 2,7 \pmod 8$ and $v \equiv 2,4 \pmod 6$, convert $\{a_{w-1},6t,6t+1\}$ to a kite using an edge of the K_w, and partition the $K_w \setminus K_2$ into kites and 4-cycles. When $w \in \{2,4\}$, remove edges $\{a_0,0\}$ and $\{a_1,0\}$ from their kites, and partition K_w together with these edges.

Case E2. $v \equiv 2 \pmod 6$ **and** $w = \frac{v+4}{6}$**.** Choose m as large as possible so that $m \leq \frac{v}{2}$, $m \leq \binom{w}{2}$, and $\binom{w}{2} - m \equiv 0 \pmod 4$. Partition the $\binom{w}{2}$ edges on W into sets E_c and E_o with $|E_c| = m$, so that the edges on E_o can be partitioned into kites and 4-cycles; this is easily

done. Place these kites and 4-cycles on W. Then let $\{e_i : 0 \le i < m\}$ be the edges in E_c; let $a_{f_i} \in e_i$ when $0 \le i < m$; $f_i = 0$ when $m \le i < \frac{v-2}{2}$; and $f_{(v-2)/2} = 1$ if $m < \frac{v}{2}$. Next form a 3-GDD of type $2^{v/2}$ on V so that $\{\{2i, 2i+1\} : 0 \le i < \frac{v}{2}\}$ forms the groups, and $\mathcal{B}$ forms the blocks. For $0 \le i < \frac{v}{2}$, let $D_{2i} = D_{2i+1} = \{0, \ldots, w-1\} \setminus \{f_i\}$. Apply the general prescription. Now for $0 \le i < \frac{v}{2}$, form the triangle $(a_{f_i}, 2i, 2i+1)$ and for $0 \le i < m$ add edge e_i to form a kite. At most three triangles remain *except when* $v \in \{14, 20\}$, where four triangles remain. To treat these cases, we reduce the number of triangles; without loss of generality, the 3-GDD contains a triple $\{v-8, v-6, v-4\}$ in a kite with edge $\{a_1, v-8\}$. Remove this kite, and form kites $(a_0, v-7, v-8; v-6)$, $(a_0, v-5, v-6; v-4)$, $(a_0, v-3, v-4; v-8)$, and $(v-2, v-1, a_1; v-8)$. □

Corollary 4.1 *Let $v \ge 4$ and $\mu_3(v)$ be defined by:*

v	6	$6t, t \ge 2$	$1+6t$	$2+6t$	9	$3+6t, t \ge 2$	4	10	$4+6t, t \ge 2$	$5+6t$
$\mu_3(v)$	1	$1+t$	t	$1+t$	1	$1+t$	1	2	$2+t$	$1+t$

Then wavecost $\mathcal{MON}(v+w, v; 4, 3) = \left\lceil \frac{(v+w)(v+w-1)}{8} \right\rceil$ if and only if $w \ge \mu_3(v)$.

4.6 Some Small Constructions

4.6.1 Used in the proof of Theorem 4.2

$\mathcal{MON}(3+3, 3; 4, 1)$: $\mathcal{B} = \{(0, a_0, 1; a_2), (1, a_1, 2; a_0), (2, a_2, 0; a_1), (a_0, a_1, a_2)\}$.

$\mathcal{MON}(4+4, 4; 4, 1)$: $\mathcal{B} = \{(1, 2, a_3; a_0), (0, 3, a_2; a_1), (a_1, 1, 3; a_0), (a_0, a_2, 1; 0), (a_0, a_1, 2; 0), (a_1, a_3, 0; a_0), (2, 3, a_3, a_2)\}$.

$\mathcal{MON}(5+5, 5; 4, 1)$: $\mathcal{B} = \{(1, 2, a_3; a_0), (0, 3, a_2; a_1), (a_1, 1, 3; a_0), (a_0, a_2, 1; 0), (a_0, a_1, 2; 0), (a_1, a_3, 0; a_0), (2, a_2, 4; a_4), (3, a_3, 4), (a_2, a_3, a_4), (2, 3, a_4), (0, 4, a_0, a_4), (1, 4, a_1, a_4)\}$.

$\mathcal{MON}(6+6, 6; 4, 1)$: $\mathcal{B} = \{(1, 2, a_3; a_0), (0, 3, a_2; a_1), (a_1, 1, 3; a_0), (a_0, a_2, 1; 0), (a_0, a_1, 2; 0), (a_1, a_3, 0; a_0), (4, 5, a_5; a_4), (2, a_2, 4; a_4), (2, 3, a_4; 5), (3, 4, a_3), (a_2, a_3, a_4), (0, 4, a_0, a_4), (1, 4, a_1, a_4), (0, 5, a_0, a_5), (1, 5, a_1, a_5), (2, 5, a_2, a_5), (3, 5, a_3, a_5)\}$.

4.6.2 Used in the proof of Theorem 4.3

$\mathcal{MON}(1+2, 1; 4, 1)$: $\mathcal{B} = \{(0, a_0, a_1)\}$.

$\mathcal{MON}(2+3, 2; 4, 1)$: $\mathcal{B} = \{(0, a_0, a_1), (1, a_1, a_2), (0, 1, a_0, a_2)\}$.

$\mathcal{MON}(3+4, 3; 4, 1)$: $\mathcal{B} = \{(0, a_0, a_1), (1, a_1, a_2), (0, 1, a_0, a_2), (2, a_2, a_3), (0, 2, a_0, a_3), (1, 2, a_1, a_3)\}$.

$\mathcal{MON}(4+5, 4; 4, 1)$: $\mathcal{B} = \{(0, 1, a_0; a_3), (0, 2, a_1; a_3), (0, 3, a_2; a_3), (2, 3, a_0; a_4), (1, 3, a_1; a_4), (1, 2, a_3; 3), (0, a_3, a_4; 3), (1, a_2, a_4; 2), (a_0, a_1, a_2; 2)\}$.

4.6.3 Used in the proof of Theorem 4.6

$\mathscr{ON}(8,4)$ with 4 triangles: $\mathcal{B} = \{(1,2,0;4), (0,3,6;7), (0,7,5;2), (4,5,3;1), (1,4,7), (1,5,6), (2,3,7), (2,4,6)\}$.

$\mathscr{MON}(7+4,7;4,2)$: $\mathcal{B} = \{(a_0,4,2;3), (a_0,3,6;0), (a_0,0,5;1), (a_1,5,3;4), (a_1,4,6;1), (a_1,1,0;2), (a_2,0,4;5), (a_2,6,5;a_3), (a_2,1,2;5), (0,3,a_3;2), (1,a_0,a_2,3), (a_0,a_1,a_2,a_3), (a_1,2,6,a_3), (1,4,a_3)\}$.

4.6.4 Used in the proof of Theorem 4.11

$\mathscr{MON}(13+3,13;4,3)$: $\mathcal{B} = \{(5+i,4+i,1+i;a_1) \mid i = 0,1,\ldots,9\} \cup \{(1+i,5+i,4+i;a_0) \mid i = 10,11,12\} \cup \{(3+i,1+i,9+i;a_2) \mid i = 6,7,\ldots,12\} \cup \{(9+i,1+i,3+i;a_0) \mid i = 1,2,\ldots,5\} \cup \{(9,3,1;a_2), (0,a_1,a_2;12), (12,a_1,a_0;0), (a_0,9,a_2,10), (a_0,a_2,11;a_1), (a_0,9,a_2,10)\}$, where the sums are computed modulo 13.

$\mathscr{MON}(15+4,15;4,3)$: $\mathcal{B} = \{(1,2,3), (a_0,4,a_1,5), (a_0,10,a_1,11), (5,4,1;a_3), (7,1,6;a_1), (6,4,2;a_3), (7,5,2;a_2), (4,7,3;a_2), (6,5,3;a_3), (9,1,8;a_1), (10,1,14;a_0), (11,1,0;a_2), (13,1,12;a_2), (10,2,8;a_2), (11,2,9;a_0), (12,2,14;a_2), (0,2,13;a_3), (8,3,11;a_3), (10,3,12;a_0), (13,3,9;a_2), (14,3,0;a_3), (12,8,4;a_2), (11,4,13;a_0), (0,10,4;a_3), (9,4,14;a_1), (8,5,13;a_2), (0,5,12;a_3), (14,11,5;a_3), (10,9,5;a_2), (8,6,0;a_0), (14,13,6;a_3), (9,6,12;a_1), (10,6,11;a_2), (14,8,7;a_3), (9,7,0;a_1), (10,7,13;a_1), (12,11,7;a_0), (6,a_2,a_0;2), (7,a_2,a_1;3), (10,a_3,a_2;1), (1,a_0,a_1;2), (9,a_1,a_3;14), (8,a_3,a_0;3)\}$.

$\mathscr{MON}(17+3,17;4,3)$: $\mathcal{B} = \{(7,16,0), (a_0,a_2,0), (a_0,1,2;3), (a_0,3,4;1), (4,5,2;a_1), (1,3,5;a_0), (16,a_0,a_1;a_2), (6,10,1;a_1), (9,14,1;a_2), (15,1,7;a_2), (1,8,12;a_2), (1,0,13;a_2), (1,16,11;a_1), (2,11,6;a_1), (2,16,8;a_2), (10,15,2;a_2), (9,2,13;a_1), (0,2,12;a_1), (2,7,14;a_2), (6,13,3;a_1), (11,3,7;a_1), (12,3,16;a_2), (9,0,3;a_2), (3,10,14;a_1), (8,3,15;a_1), (14,6,4;a_2), (4,11,15;a_2), (7,12,4;a_1), (13,4,8;a_1), (4,16,9;a_2), (0,4,10;a_1), (5,12,6;a_2), (7,13,5;a_2), (8,14,5;a_1), (15,5,9;a_1), (5,16,10;a_2), (5,0,11;a_2), (9,7,6;a_0), (10,8,7;a_0), (11,9,8;a_0), (12,10,9;a_0), (13,11,10;a_0), (14,12,11;a_0), (15,13,12;a_0), (16,14,13;a_0), (0,15,14;a_0), (6,16,15;a_0), (8,6,0;a_1)\}$.

$\mathscr{MON}(17+4,17;4,3)$: $\mathcal{B} = \{(2,9,11), (9,12,16), (a_0,13,14;15), (a_0,15,16;13), (16,0,14;a_1), (13,15,0;a_0), (13,2,1;a_3), (13,12,3;a_3), (13,11,4;a_3), (5,10,13;a_1), (6,9,13;a_2), (7,8,13;a_3), (14,4,2;a_3), (14,12,5;a_3), (11,14,6;a_3), (14,10,7;a_3), (1,3,14;a_2), (9,8,14;a_3), (1,4,15;a_1), (3,5,15;a_2), (2,6,15;a_3), (15,7,12;a_3), (15,11,8;a_3), (1,16,5;a_1), (6,4,16;a_2), (3,7,16;a_3), (2,8,16;a_1), (10,16,11;a_3), (1,6,0;a_1), (4,8,0;a_2), (10,15,9,;a_3), (2,10,0;a_3), (5,0,7;a_1), (3,0,9;a_1), (12,0,11;a_1), (1,a_0,7;6), (8,6,a_0;a_3), (9,a_0,5;11), (10,a_0,4;9), (11,a_0,3;10), (2,a_0,12;8), (8,a_1,1;11), (10,a_1,6;3), (12,4,a_1;a_3), (3,a_1,2;7), (1,a_2,9;7), (10,a_2,8;3), (11,a_2,7;4), (12,a_2,6;5), (2,a_2,5;8), (3,a_2,4;5), (a_1,a_0,a_2;a_3), (12,1,10;a_3)\}$.

4.7 Conclusions

The determination of $cost\,\mathscr{O}\mathscr{N}(n, v; C, C')$ appears to be easier when $C' = 4$ than the case for $C' = 3$ settled in [80, 81]. Nevertheless the very flexibility in choosing kites, 4-cycles, or triangles also results in a wide range of numbers of wavelengths among decompositions with optimal drop cost. This leads naturally to the question of minimizing the drop cost and the number of wavelengths simultaneously. In many cases, the minima for both can be realized by a single decomposition. However, it may happen that the two minimization criteria compete. Therefore we have determined the minimum number of wavelengths among all decompositions of lowest drop cost for the specified values of n, v, and C'.

Part III

Degree-constrained Subgraphs

III.1 Motivation

We begin with an example. Let G be a 4-regular (multi)graph. Does G contain a 3-regular subgraph? Not necessarily, as the example of a triangle with two edges between each pair of vertices shows. Now suppose we add an arbitrary edge to G. It turns out that the claim is now true.

Theorem III.1 (Alon, Friedland, and Kalai [32]) *Every 4-regular graph plus an edge contains a 3-regular subgraph.*

Theorem III.1 was proved using facts from number theory. Using more sophisticated arguments, the same authors proved the following strengthening of Theorem III.1.

Theorem III.2 (Alon, Friedland, and Kalai [33]) *Suppose that every vertex of a graph G has degree k or $k+1$, and at least one vertex has degree $k+1$. Then for every prime power q such that $k \geq 2(q-1)$, G contains a q-regular subgraph.*

The above examples illustrate how apparently simple but difficult may be to assure the existence of a subgraph satisfying certain degree constraints. A general instance of a Degree-constrained Subgraph problem consists of an edge-weighted or vertex-weighted graph G and the objective is to find an optimal weighted subgraph, subject to certain degree constraints on the vertices of the subgraph. This class of combinatorial problems has been extensively studied due to its numerous applications in network design, as it is the case of the Minimum-degree Spanning Tree [130] and the Minimum-degree Steiner Tree [131] problems. If the input graph is bipartite, they have as particular cases classical transportation and assignment problems in operations research [85]. These problems have attracted a lot of attention in the last decades and have resulted in a large body of literature [30, 38, 54, 60, 61, 65, 111, 112, 112, 130–132, 156, 166, 168, 172, 181, 193].

In the sequel of the second part of this thesis we shall convince the reader that this family of problems provides a rich source of nice and interesting problems to apply a variety of algorithmic techniques and graph-theoretical approaches.

Relation to traffic grooming. We now discuss how degree-constrained subgraph problems are related to traffic grooming. More precisely, how the study of traffic grooming lead the author to the study of degree-constrained subgraph problems.

We have presented in Section 1.4 (page 49) an algorithm that computes a solution to the Ring Traffic Grooming problem with an approximation ratio of $O(n^{1/3} \cdot \log^2 n)$, where n is the size of the ring. This approximation ratio can be informally factorized as follows.

- a factor $\log n$ comes from the fact of partitioning the request set into $\log n$ classes of similar length;
- another factor $\log n$ comes from the fact of greedily removing subgraphs from the request graph[1];

[1] It is usual to *pay* a factor $\log n$ in greedy algorithms for NP-hard problems, the archetypical example being the $\log n$-approximation for the Minimum Set Cover problem [202].

- finally, the factor $n^{1/3}$ appears because we used the algorithm of [114] for the DENSE k-SUBGRAPH (DkS) problem (see page 26) to find dense subgraphs.

That is, if we had an approximation algorithm with ratio β for the DkS problem, this would automatically yield an approximation algorithm with ratio $O(\beta \cdot \log^2 n)$ for the RING TRAFFIC GROOMING problem.

Unfortunately, the DkS problem is a really difficult problem. First of all, note that the NP-hardness of DkS easily follows from the NP-hardness of MAXIMUM CLIQUE. On the other hand, if we do not fix the size of the subset of vertices S, then finding a densest subgraph of G reduces to an instance of the MAX-FLOW MIN-CUT problem, and hence it can be solved in polynomial time (see [163, Chapter 4] for more details). The DkS problem has attracted a lot of attention during the last decade, primarily in approximation algorithms [92, 114, 159]. Understanding the complexity of DkS remains widely open, as the gap between the best hardness result (APX-hardness [159]) and the best approximation algorithm (with ratio $O(n^{1/3-\varepsilon})$ [114]) is huge. Andersen and Chellapilla have recently studied in [37] two relaxed versions of the DkS problem. Namely, they have proved that the problem of finding a densest subgraph with *at least* k vertices can be efficiently approximated (they provide a 3-approximation), but that the problem of finding a densest subgraph with *at most* k vertices is essentially as hard as the DkS problem itself.

Therefore, it seems natural to ask whether one can efficiently find *dense* subgraphs using another approach. As it seems to be very hard to find subgraphs with the greatest density, or equivalently with the greatest *average degree*, one could try to find small subgraphs with prescribed *minimum degree*, which would also guarantee high density. This discussion naturally leads to the definition of the following problem (see Chapter 6 for more details).

$\leq k$-SIZE SUBGRAPH OF MINIMUM DEGREE $\geq d$ (kSMDd)
Input: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does there exist a subset $S \subseteq V$, with $\lvert S \rvert \leq k$, such that $G[S]$ has minimum degree at least d?

Let us see how kSMDd is related to DkS. Suppose we are looking for an induced subgraph $G[S]$ of size at most k and with density at least ρ. In addition, assume that S is minimal, i.e., no subset of S has density greater than $\rho(S)$. This implies that every vertex of S has degree at least $\rho/2$ in $G[S]$. To see this, observe that if there is a vertex v with degree strictly smaller than $\rho/2$, then removing v from S results in a subgraph of density greater than $\rho(S)$ and of smaller size, contradicting the minimality of S. Secondly, if we have an induced subgraph $G[S]$ of minimum degree at least ρ, then S is a subset of density at least $\rho/2$. These two observations together show that, modulo a constant factor, looking for a densest subgraph of G of size at most k is equivalent to looking for the largest possible value of ρ for which kSMDρ returns YES for the parameter k. As the local degree conditions are more rigid than the global density of a subgraph, it may be more convenient to work directly with kSMDd. A better understanding of the kSMDd problem could provide an alternative way to approach the DkS problem, and therefore by transitivity also to approach the TRAFFIC GROOMING problem.

III.2 State-of-the-art and our Contribution

In the preceding section we have introduced the problem of finding a minimum subgraph with given minimum degree. It is natural to consider its dual version: finding a maximum subgraph (in terms of number of edges or vertices) with given maximum degree. These two problems and some variants are the main topic of Chapters 5, 6, and 7.

III.2.1 The role of connectivity

As mentioned above, we will study in the forthcoming chapters the problem of finding a maximum subgraph with given maximum degree. If we do not impose any further restriction on the desired subgraph, it turns out that the problem (for both the vertex and edge version) is solvable in polynomial time using matching techniques [85, 134, 165]. In fact, also the version where nodes or edges have associated weights and the objective is to maximize the total weight of a subgraph respecting the degree constraints, is solvable in polynomial time [134] (as it happens for the classical Maximum Matching problem). Even a more general version where the input contains an interval of allowed degrees for each node is known to be solvable in polynomial time [165].

Suppose now that we restrict the output subgraph to be connected, i.e., we would like to find a maximum *connected* subgraph with given maximum degree. Then the problem is exactly one of the classical NP-hard problems of Garey and Johnson's book: [134, Problem GT26]. Is this phenomenon surprising? Why does connectivity change completely the complexity of the problem? The explanation is the following: the degree of a vertex in a solution is a *local* property, in the sense that it depends only on how many of its incident edges are included in the solution; on the other hand, connectivity is a *global* property, as in order to check that a subgraph is connected one needs to explore the whole subgraph. Therefore, it seems natural that if the only constraints on the solution are the degree of the vertices, an algorithm along the lines of the classical augmenting paths algorithm to find a maximum matching (see for instance [165]) could find an optimal solution in polynomial time. This change from *local* to *global* is what makes the problem become NP-hard.

We prove in Chapter 5 that the problem is in fact even harder: it is not in Apx (see page 23) for any fixed maximum degree $d \geq 2$ unless P≠NP, and if there is a polynomial time algorithm for $d \geq 2$, with performance ratio $2^{O(\sqrt{\log n})}$, then NP $\subseteq$ DTIME($2^{O(\log^5 n)}$). These results were previously known only for the case $d = 2$ [156], that corresponds to the Longest Path problem. On the other hand, we give an approximation algorithm for general unweighted graphs with ratio $\min\{m/\log n,\ nd/(2\log n)\}$, and an approximation algorithm for general weighted graphs with ratio $\min\{n/2,\ m/d\}$. (Here $n = |V(G)|$ and $m = |E(G)|$, where G is the input graph.) The first algorithm uses an algorithm introduced in [35], which is based on the *color-coding* method. We also present a constant-factor approximation when the input graph has a low-degree spanning tree, in terms of the integer d.

We also prove in Chapter 5 that the problem of finding a minimum subgraph of given minimum degree is not in Apx for any fixed degree $d \geq 3$. These are the first hardness results for this problem.

To prove the hardness results for the two problems mentioned above, we use the error amplification technique, which is a powerful tool to prove that certain approximation algorithms cannot exist provided, as usual, that P≠NP.

III.2.2 Parameterizing the input

In order to better understand the complexity of NP-hard problems, the theory of parameterized complexity (see page 24) provides a framework to refine the measure of the complexity of such problems, by means of introducing a parameter to the input (which is commonly assumed to be small compared to the size of the input) and analyzing the running time of the algorithms in terms of that parameter. Since the degree-constrained subgraph problems we consider in this part of the thesis seem to be really hard to solve – in view of the inapproximability results of Chapter 5 –, the next natural step is to analyze their parameterized complexity. This is the topic of Chapter 6.

This approach brings us a very interesting insight. Indeed, when parameterizing by the size of the desired subgraph, the behavior of the two aforementioned problems is completely different, whereas the seem to have similar (classical) complexity. Namely, the problem of finding a maximum connected subgraph with bounded maximum degree is FPT, and the problem of finding a minimum subgraph (that of course, we can assume to be connected) with bounded minimum degree is $W[1]$-hard. The fact that the first problem is FPT follows directly from the fact that the corresponding parameter is minor closed (see page 25 for the definition of minor closed parameters), as it is proved in Lemma 7.2 (page 173). Proving the $W[1]$-hardness of the second problem is a more difficult task; see Chapter 6 for the details. Moreover, we prove in Chapter 6 that the problem of finding a minimum regular subgraph (induced or not) is $W[1]$-hard, by a reduction from the Multi-Color Clique problem (see Theorems 6.1 and 6.2 in Chapter 6).

III.2.3 Topologically restricting the input

A class of graphs is called *sparse* if for every graph G in this class, $|E(G)| = O(|V(G)|)$. Examples of sparse graph classes are planar graphs (for which the number of edges is at most three times the number of edges [97]), graphs embeddable in a fixed surface or, more generally, families of graphs excluding a fixed graph as a minor. When restricting the input graphs to belong to some sparse class, it is usual that *hard* problems become *easier*. This paradigm is the cornerstone idea of the second part of Chapter 6 and Chapters 7 and 8.

Namely, in Section 6.3 we provide explicit FPT algorithms for the discussed problems in graphs with bounded local treewidth and graphs with excluded minors. These algorithms are based on dynamic programming techniques and structural results from graph minors theory. We stress that the *classification result* asserting that these problems are FPT for graphs with excluded minors is a consequence of the powerful meta-theorems of Frick and Grohe [128] and of Dawar, Grohe, and Kreutzer [89] (see page 156 of Section 6.1.3 for the details). Our contribution is to give explicit algorithms that run considerably faster than those implied by the mentioned meta-theorems.

As mentioned in Section III.2.2, the fact that the problem of finding a maximum connected degree-bounded subgraph is FPT is a direct consequence of the Graph Minors Theorem. In general graphs, one cannot hope the function $f(k)$ to be subexponential (i.e., $2^{o(k)}$), because this would contradict the widely believed exponential time hypothesis [101]. However, when we restrict the input to be planar, then such algorithms are indeed possible. In Chapter 7 we obtain subexponential parameterized algorithms for the mentioned problem, as well as for a few variants. Our approach follows the general framework that has been developed during the last years, which combines bidimensionality theory and refined dynamic programming techniques [95, 98–102, 144]. As it is usual in bidimensionality theory, we obtain algorithms with $f(k) = 2^{O(\sqrt{k})}$. Loosely speaking, the idea of this approach is as follows. Suppose we have a graph G with branchwidth $\mathbf{bw}(G)$ and we want to decide whether a parameter $\mathbf{P}$ is at least k in G (for instance, "does G have a path of length at least k?"). We distinguish two cases according to $\mathbf{bw}(G)$:

- If $\mathbf{bw}(G)$ is *big* (greater than $\sqrt{k}$), we must exhibit a *certificate* that $\mathbf{P}(G)$ is also *big*, by proving that the parameter is minor closed and looking at its behavior on the square grid. Then the answer to the parameterized problem is automatically YES.
- Otherwise, if $\mathbf{bw}(G)$ is *small* (smaller than $\sqrt{k}$), we compute $\mathbf{P}(G)$ efficiently using Catalan structures and dynamic programming techniques over an optimal branch decomposition of G.

Finally, we deal in Chapter 8 with graphs embedded in surfaces [171]. In this case we consider a more general family of problems than the ones considered so far, as specified in the paragraph below. It is a common approach for solving NP-hard problems to use dynamic programming algorithms over a branch (or tree) decomposition of the input graph [58]. We focus on this approach, and particularize it when the input graph is embedded in a surface.

Namely, we provide in Chapter 8 a framework for the design of $2^{O(k)} \cdot n$ step dynamic programming algorithms for surface-embedded graphs on n vertices of branchwidth at most k. Our technique applies to graph problems for which dynamic programming uses tables encoding set partitions. For general graphs, the best known algorithms for such problems run in $2^{O(k \cdot \log k)} \cdot n$ steps. That way, we considerably extend the class of problems that can be solved by algorithms whose running times have a *single exponential dependence* on branchwidth, and improve the running time of several existing algorithms. Our approach is based on a new type of branch decomposition called *surface cut decomposition*, which generalizes sphere cut decompositions for planar graphs (see page 176), and where dynamic programming should be applied for each particular problem. The construction of such a decomposition uses a new graph-topological tool called *polyhedral decomposition*. The main result of this chapter – namely, Theorem 8.4 in page 216 – is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is *single exponential*. This fact is proved by a detailed analysis of how non-crossing partitions are arranged on surfaces with boundary and uses diverse techniques from topological graph theory and analytic combinatorics. As a consequence of our results, we can derive subexponential exact and parameterized algorithms in graphs of bounded genus for the problems in the mentioned category.

Chapter 5

Hardness and Approximation

An instance of a Degree-Constrained Subgraph problem consists of an edge-weighted or vertex-weighted graph $G = (V, E)$, $|V| = n$, $|E| = m$, and the objective is to find an optimal weighted subgraph, subject to certain degree constraints on the vertices of the subgraph. This chapter considers three natural Degree-Constrained Subgraph problems and studies their approximability.

The Maximum Degree-Bounded Connected Subgraph (MDBCS$_d$) problem takes as additional input a weight function $\omega : E \to \mathbb{R}^+$ and an integer $d \geq 2$, and asks for a subset $E' \subseteq E$ such that the subgraph $G' = (V, E')$ is connected, has maximum degree at most d, and $\sum_{e \in E'} \omega(e)$ is maximized. We prove that MDBCS$_d$ is not in Apx for any $d \geq 2$, and that if there is a polynomial time algorithm for MDBCS$_d$, $d \geq 2$, with performance ratio $2^{\mathcal{O}(\sqrt{\log n})}$, then NP $\subseteq$ DTIME($2^{\mathcal{O}(\log^5 n)}$). On the other hand, we provide a $(\min\{m/\log n,\ nd/(2\log n)\})$-approximation algorithm for unweighted graphs, and a $(\min\{n/2,\ m/d\})$-approximation algorithm for weighted graphs. We also prove that when G has a low-degree spanning tree, the MDBCS$_d$ problem can be approximated within a small constant factor in unweighted graphs.

The Minimum Subgraph of Minimum Degree$_{\geq d}$ (MSMD$_d$) problem involves finding a smallest subgraph of G with minimum degree at least d. We prove that MSMD$_d$ is not in Apx for any $d \geq 3$ and provide an $\mathcal{O}(n/\log n)$-approximation algorithm for the classes of graphs excluding a fixed graph as a minor, using dynamic programming techniques and a known structural result on graph minors. In particular, this approximation algorithm applies to planar graphs and graphs of bounded genus.

The Dual Degree-Dense k-Subgraph (DDDkS) problem involves finding a subgraph H of G such that $|V(H)| \leq k$ and δ_H is maximized, where δ_H is the minimum degree in H. We present a deterministic $\mathcal{O}(n^\delta)$-approximation algorithm in general graphs, for some universal constant $\delta < 1/3$.

Keywords: approximation algorithms, degree-constrained subgraphs, hardness of approximation, Apx, PTAS, dense subgraphs, graph minors, excluded minor.

5.1 Introduction

In this chapter we consider three natural Degree-Constrained Subgraph problems and study them in terms of approximation algorithms. A general instance of a Degree-Constrained Subgraph problem [30, 38, 193] consists of an edge-weighted or vertex-weighted graph and the objective is to find an optimal weighted subgraph, subject to certain degree constraints on the vertices of the subgraph.

Degree-Constrained Subgraph problems have attracted a lot of attention in the last decades and have resulted in a large body of literature [30, 38, 112, 130–132, 156, 166, 181, 193]. The most well-studied ones are probably the Minimum-Degree Spanning Tree [130] and the Minimum-Degree Steiner Tree [131] problems.

Beyond the esthetic and theoretical appeal of Degree-Constrained Subgraph problems, the reasons for such intensive study are rooted in their wide applicability in the areas of interconnection networks and routing algorithms, among others. For instance, given an interconnection network modeled by an undirected graph, one may be interested in finding a small subset of nodes having a high degree of connectivity for each node. This translates into finding a small subgraph with a lower bound on the degree of its vertices, i.e., to the MSMD_d problem, to be defined shortly. Note that if the input graph is bipartite, these problems are equivalent to classical transportation and assignment problems in operation research.

The first problem studied in this chapter is a classical NP-hard problem listed in [134] (c.f. Problem [GT26] for the unweighted version). Let $d \geq 2$ be a fixed integer.

Maximum Degree-Bounded Connected Subgraph (MDBCS_d)
Input: A graph $G = (V, E)$ and a weight function $\omega : E \to \mathbb{R}^+$.
Output: A subset $E' \subseteq E$ such that the subgraph $G' = (V, E')$ is connected (except, possibly, for isolated vertices), has maximum degree at most d, and $\sum_{e \in E'} \omega(e)$ is maximized.

For $d = 2$, the unweighted MDBCS_d problem corresponds to the Longest Path problem. Indeed, given the input graph G (which can be assumed to be connected), let P and G' be optimal solutions of Longest Path and MDBCS_2 in G, respectively. Then observe that $|E(G')| = |E(P)|$ unless G is Hamiltonian, in which case $|E(G')| = |E(P)| + 1$. One could also ask the question: what happens when G' is not required to be connected in the definition of MDBCS_d? It turns out that without the connectivity constraint, both the edge version and the vertex version (where the goal is to maximize the total weight of the vertices of a subgraph respecting the degree constraints) of the MDBCS_d problem are known to be solvable in polynomial time using matching techniques [85, 134, 165]. In fact, without connectivity constraints, even a more general version where the input contains an interval of allowed degrees for each node is known to be solvable in polynomial time.

The most general version of Degree-Constrained Subgraph problem is to find a subgraph under constraints given by lower and upper bounds on the degree of each vertex, the objective being to minimize or maximize some parameter (usually the size of the subgraph). A common variant ignores the lower bound on the degree and just requires

the vertices of the subgraphs to have a given maximum degree [181], in which case the typical optimization criterion is to maximize the size of a subgraph satisfying the degree constraints. The resulting problem is also called an Upper Degree-Constrained Subgraph problem in [132]. In contrast, we are unaware of existing results considering just a lower bound on the degrees of the vertices of the subgraph, except for combinatorial conditions on the existence of such a subgraph [111,112]. In an attempt to fill this void in the literature, the last two problems considered in this chapter aim at minimizing the size of a subgraph and maximizing the lower bound on the minimum degree, respectively. For a graph H, let δ_H denote the minimum degree of the vertices in H. Let $d \geq 2$ be fixed.

Minimum Subgraph of Minimum Degree$_{\geq d}$ (MSMD$_d$)
Input: An undirected graph $G = (V, E)$.
Output: A subset $S \subseteq V$ such that for $H = G[S]$, $\delta_H \geq d$ and $\lvert S \rvert$ is minimized.

Dual Degree-Dense k-Subgraph (DDDkS)
Input: An undirected graph $G = (V, E)$ and a positive integer k.
Output: An induced subgraph H of size $\lvert V(H) \rvert \leq k$, such that δ_H is maximized.

Note that the problem kSMDd defined in Section III.1 (page 124) is the parameterized version of the MSMD$_d$ problem defined above. MSMD$_d$ is closely related to MDBCS$_d$. Indeed, MSMD$_d$ corresponds exactly to the dual (unweighted) node-minimization version of MDBCS$_d$. MSMD$_d$ is also a generalization of the Girth problem (finding a shortest cycle), which corresponds exactly to the case $d = 2$. The MSMD$_d$ is studied in the realm of parameterized complexity in Chapter 6, where it is shown that MSMD$_d$ is W[1]-hard for $d \geq 4$ and explicit *fixed-parameter tractable* (FPT) algorithms are given for the class of graphs excluding a fixed graph as a minor and graphs of bounded local treewidth, for $d \geq 3$. Besides the above discussion, our main motivation for studying MSMD$_d$ is its close relation to the well studied Dense k-Subgraph (DkS) [114, 159] and Traffic Grooming [J1] problems. See Section III.1 (page 123) for more details.

The above discussion illustrates that the study of the above mentioned problems is very natural and that the results obtained for them can reverberate in several other important optimization problems, coming from both theoretical and practical domains.

Our Contribution: In this chapter we obtain both approximation algorithms and results on hardness of approximation. All of our hardness results are based on the hypothesis that P $\neq$ NP. More precisely, our results are the following:

- We prove that the MDBCS$_d$ problem is not in Apx for any $d \geq 2$, and that if there is a polynomial time algorithm for MDBCS$_d$, $d \geq 2$, with performance ratio $2^{O(\sqrt{\log n})}$, then NP $\subseteq$ DTIME($2^{O(\log^5 n)}$). On the other hand, we give an approximation algorithm for general unweighted graphs with ratio $\min\{m/\log n,\ nd/(2\log n)\}$, and an approximation algorithm for general weighted graphs with ratio $\min\{n/2,\ m/d\}$. The first algorithm uses an algorithm introduced in [35], which is based on the *color-coding* method. We also present a constant-factor approximation when the input graph has a low-degree spanning tree, in terms of the integer d.

- We prove that the MSMD$_d$ problem is not in Apx for all $d \geq 3$. The proof is obtained by the following two steps. First, by a reduction from Vertex Cover, we prove that MSMD$_d$ does not admit a PTAS. In particular, this implies that MSMD$_d$ is NP-hard for any $d \geq 3$. Then, we use the *error amplification* technique to prove that MSMD$_d$ is not in Apx for any $d \geq 3$. On the positive side, we give an $O(n/\log n)$-approximation algorithm for the class of graphs excluding a fixed graph H as a minor, using a known structural result on graph minors and dynamic programming over graphs of bounded treewidth. In particular, this gives an $O(n/\log n)$-approximation algorithm for planar graphs and graphs of bounded genus.
- We give a deterministic $O(n^{\delta})$-approximation algorithm for the DDDkS problem in general graphs, for some universal constant $\delta < 1/3$. We also provide a randomized $O(\sqrt{n}\log n)$-approximation algorithm, which is completely different in nature. Although the approximation ratio is significantly worse, the idea of the proof is quite simple and nice.

Organization of the chapter: The basic definitions required in this chapter can be found in Section I.2 (page 23). In Section 5.2 we establish that MDBCS$_d$ is not in Apx for any $d \geq 2$, and in Section 5.3 we present two approximation algorithms for unweighted and weighted general graphs, respectively. The constant-factor approximation for MDBCS$_d$ when the input graph has a low-degree spanning tree is provided in Section 5.3.2 for unweighted graphs. In Section 5.4 we prove that MSMD$_d$ is not in Apx for any $d \geq 3$, and in Section 5.5 we give an $O(n/\log n)$-approximation algorithm for the class of graphs excluding a fixed graph H as a minor. In Section 5.6 we give two approximation algorithms for the DDDkS problem. Finally, we conclude with some remarks and open problems in Section 5.7.

5.2 Hardness of Approximating MDBCS$_d$

As mentioned in Section 7.1, MDBCS$_2$ corresponds to the Longest Path (or Cycle) problem, which is known to not admit any constant-factor approximation [156], unless P = NP. In this section we extend this result and prove that, under the assumption that P≠NP, MDBCS$_d$ is not in Apx for any $d \geq 2$, proving first that MDBCS$_d$ is not in PTAS for any $d \geq 2$. Finally, we also prove in Theorem 5.4 that if there is a polynomial time algorithm for MDBCS$_d$, $d \geq 2$, with a performance ratio of $2^{O(\sqrt{\log n})}$, then NP $\subseteq$ DTIME($2^{O(\log^5 n)}$).

Theorem 5.1 MDBCS$_d$ *does not admit a* PTAS *for any* $d \geq 2$*, unless* P = NP*.*

Proof: We prove the result for the case when $d \geq 3$; the result for $d = 2$ follows from [156]. We give our reduction from the *traveling salesman problem* TSP(1, 2), which does not admit a PTAS unless P = NP [176]. An instance of TSP(1, 2) consists of a complete graph $G = (V, E)$ on n vertices and a weight function $f : E \to \{1, 2\}$ on its edges, and the objective is to find a traveling salesman tour of minimum weight in G.

We show that if there is a PTAS for MDBCS$_d$, $d \geq 3$, then one can construct a PTAS for TSP$(1,2)$. Towards this, we transform the graph G into a new augmented graph G' with a modified weight function g on its edges. For every vertex $v \in V$ we add to G' $d-2$ new vertices $\{v_1, \cdots, v_{d-2}\}$ and an edge from v to every vertex v_i, $1 \leq i \leq d-2$. Thus $G' = (V \cup V', E \cup E')$ where $V' = \bigcup_{v \in V}\{v_1, \cdots, v_{d-2}\}$ is the set of new vertices and $E' = \{\{v_i, v\} \mid 1 \leq i \leq d-2, v \in V\}$ is the set of new edges. We define the weight function g on the edges of G' as:

$$g(e) = \begin{cases} 3 - f(e), & e \in E, \quad \text{(weights of original edges get flipped)} \\ 3, & e \in E'. \end{cases}$$

Next we prove a claim characterizing the structure of the *maximal* solutions of MDBCS$_d$ in G'. Essentially, it shows that any given solution G_1 of MDBCS$_d$ in G' with value W can be transformed into another solution G_2 of MDBCS$_d$ in G' with value at least W, such that G_2 contains all the newly added edges and induces a Hamiltonian cycle in G.

Claim 5.1 *Any given solution $G_1 = (V \cup V', E_1)$ to* MDBCS$_d$ *in G' can be transformed in polynomial time into a solution $G_2 = (V \cup V', E_2)$ of* MDBCS$_d$ *in G' such that (i) $G_3 = (V, E \cap E_2)$ is a Hamiltonian cycle in G, and (ii) $\sum_{e \in E_2} g(e) \geq \sum_{e' \in E_1} g(e')$.*

Proof: We prove the claim by describing a series of transformations, applied in order of appearance, successively improving the solution, and eventually yielding the desired G_2. For a given edge set F, let $X(F)$ be the set of vertices containing the end-points of the edges in F.

(a) Suppose $E_1 \cap E' = \emptyset$. Then $H = (X(E_1), E_1)$ is connected and every vertex $v \in X(E_1)$ has degree at most d in H. This implies that H contains a cycle, so removing any edge from this cycle will not break connectivity. So we can remove any edge $\{u, v\}$ from this cycle and add the edges $\{u_1, u\}$ and $\{v_1, v\}$, obtaining a solution of larger weight. Therefore, we assume henceforth that $E_1 \cap E' \neq \emptyset$.

(b) Suppose $V \setminus X(E_1) \neq \emptyset$, that is, there is a vertex $v \in V$ which is not contained in $X(E_1)$. In this case, by Observation (a) there exists a vertex $u \in X(E_1)$ such that one of the edges $\{u_i, u\}$, $1 \leq i \leq d-2$, is in E_1. We then set $E_1 \leftarrow E_1 - \{\{u_i, u\}\} \cup \{\{u, v\}, \{v, v_i\} \mid 1 \leq i \leq d-2\}$. Clearly, connectivity is maintained (as removing edges from E' does not break connectivity) and the weight of solution increases by at least 1. This procedure is repeated until the current solution contains all the vertices of G.

(c) Suppose $H' = (V, E \cap E_1)$ is neither a spanning tree nor a Hamiltonian cycle. Notice that H' is connected, as removing degree 1 vertices of V' does not break connectivity. This implies that there is a cycle C in H' and a vertex v on it such that $deg_{H'}(v) \geq 3$ (otherwise, H' would be disconnected). This implies that there exists an edge $e = \{v, v_i\}$ such that $e \notin E_1$. Let $\{u, v\}$ be an edge on C. We then set $E_1 \leftarrow E_1 - \{\{u, v\}\} \cup \{\{v, v_i\}\}$. Again, connectivity is clearly maintained (as removing an edge from a cycle does not break connectivity) and the weight of the solution increases by at least 1. This procedure is repeated until H' is either a spanning tree or a Hamiltonian cycle.

(d) Suppose $H' = (V, E \cap E_1)$ is a spanning tree. If H' is a path then the end-points of this path, say u and v, have degree 1 in H', hence we can add the edge $\{u, v\}$ and obtain a solution of greater weight. So let us suppose that H' is not a path, hence

there exists a vertex v of degree at least 3 in H'. This implies that there exists an edge $e = \{v, v_i\}$ such that $e \notin E_1$. Let $\{u, v\}$ be an edge incident to v in the spanning tree H'. Consider the spanning forest $H' - \{\{u, v\}\}$, consisting of two sub-tress H'_u and H'_v containing u and v respectively. Select a leaf $w_1 \in H'_u$ and a leaf $w_2 \in H'_v$ ($w_2 \neq v$), and set $E_1 \leftarrow E_1 - \{\{u, v\}\} \cup \{\{v, v_i\}, \{w_1, w_2\}\}$. Clearly the resulting graph is connected and has greater weight.

The above transformation rules can be applied in polynomial time to obtain a graph G_3 that is a solution of MDBCS$_d$ in G' and satisfies the conditions described in the statement of the claim. □

Suppose that there exists a PTAS for MDBCS$_d$ realized by an approximation scheme $\mathcal{A}_\delta$. This family of algorithms takes as input a graph G'' and a parameter $\delta > 0$, and returns a solution of MDBCS$_d$ of weight at least $(1 - \delta)OPT_{G''}$, where $OPT_{G''}$ is the value of an optimum solution of MDBCS$_d$ in G''.

Using this scheme, we now proceed to construct a PTAS for TSP$(1, 2)$. Given a graph G, an instance of TSP$(1, 2)$, and $\varepsilon > 0$, do the following.

1. Apply the transformation described before Claim 5.1 to G and obtain the graph G'.
2. Fix $\delta = h(\varepsilon, d)$ (to be specified later) and run $\mathcal{A}_\delta$ on G'. Let G'' be the resulting solution.
3. Apply the polynomial time transformation described in Claim 5.1 on G'', the solution obtained by $\mathcal{A}_\delta$ on G'. Let the new solution be $G^* = (V \cup V_1, E^*)$.
4. Return $E^* \cap E$ as the solution of TSP$(1, 2)$.

Now we prove that the solution returned by our algorithm satisfies $\sum_{e \in E^* \cap E} f(e) \leq (1+\varepsilon)O_T$, where O_T is the weight of an optimum tour in G. Let such an optimum tour contain a edges of weight 1 and b edges of weight 2. Then $O_T = a + 2b$ and $a + b = n$. Equivalently $a = 2n - O_T$ and $b = O_T - n$. Let O_D be the value of an optimum solution of MDBCS$_d$ in G'. Then by Claim 5.1 and the flipping nature of the function g, we have that

$$O_D = (d - 2)3n + 2a + b. \tag{5.1}$$

Let $3(d - 2)n + O^*_D$ be the value of the solution returned by $\mathcal{A}_\delta$, where O^*_D is the sum of the edge weights of the Hamiltonian cycle in G, that is, $O^*_D = \sum_{e \in E^* \cap E} g(e)$. Since $\mathcal{A}_\delta$ is a PTAS,

$$3(d - 2)n + O^*_D \geq (1 - \delta)O_D. \tag{5.2}$$

Combining Equation (5.1) and Inequality (5.2) gives

$$O^*_D \geq (1 - \delta)O_D - 3(d - 2)n = 3n - O_T + \delta O_T - n(3d - 3)\delta. \tag{5.3}$$

On the other hand, the value of the solution returned by our algorithm for TSP$(1, 2)$ is $O^*_T = 3n - O^*_D$ (since if $O^*_D = 2x + y$, x being the number of edges of weight 2 and y being the number of edges of weight 1, with $x + y = n$, then the value of the solution for TSP$(1, 2)$ is $x + 2y$). Substituting $O^*_D = 3n - O^*_T$ in Inequality (5.3) and using the fact that $O_T \geq n$ yields

$$O^*_T \leq O_T - \delta O_T + n(3d - 3)\delta \leq O_T - \delta n + n(3d - 3)\delta. \tag{5.4}$$

To show that $O_T^* \leq (1+\varepsilon)O_T$, by (5.4) it is enough to show that $-\delta n + n(3d-3)\delta \leq \varepsilon \cdot O_T$. Rather, we show that $-\delta n + n(3d-3)\delta \leq \varepsilon n$, which automatically implies the required bound. This can be done by setting $\delta = h(\varepsilon, d) = \frac{\varepsilon}{3d-4}$, yielding a PTAS for TSP$(1,2)$. Since TSP$(1,2)$ does not admit a PTAS [176], the last assertion also rules out the existence of a PTAS for MDBCS$_d$ for any $d \geq 3$, unless P = NP. □

To show the non-existence of a constant factor approximation, we first make a simplification and then define a graph product.

Remark 5.1 *We assume that the input graph $G = (V, E)$ of* MDBCS$_d$ *admits an optimal solution containing a cycle. This can be assumed without loss of generality by adding, for each vertex $v \in V$, two new vertices v_1, v_2 and a triangle consisting of the edges $\{v, v_1\}, \{v, v_2\}, \{v_1, v_2\}$ with weight ε, $0 < \varepsilon << \min_{e \in E} \omega(e)$. Let G' be the transformed graph. By the choice of ε, the new edges do not affect the structure of the optimal solutions in G. If all the optimal solutions in G were trees, these triangles could be added (for $d \geq 3$) to the leaves of any solution in G to obtain a solution in G' containing a cycle.*

We define the following *edge squaring* operation.

Definition 5.1 *Given a graph $G = (V, E)$ and an edge $\{u, v\} \in E$, define $G^2_{u,v}$ as the graph obtained from G by replacing every edge $e = \{x, y\} \in E$ with a copy G_e of G and adding the two edges $\{x, u\}$ and $\{y, v\}$ with weight ε, $0 < \varepsilon << \min_{e \in E} \omega(e)$. The vertices x and y are referred as the* contact vertices *of G_e.*

Observe that this graph product differs from the *edge square* graph introduced in [156] to prove the hardness of Longest Path, in which for every edge $e = \{x, y\} \in E$, the vertices x and y are joined to every vertex in G_e. We need this new definition for technical reasons, as the structure of the solutions of MDBCS$_d$ for $d = 2$ and for $d \geq 3$ differs considerably. The graphs $G^2_{u,v}$ are important because of the following lemma.

Lemma 5.1 *Let $G = (V, E)$ be a graph for which the total edge weight in an optimal solution for* MDBCS$_d$ *is ℓ. Then,*

(a) *there exists an edge $\{u, v\} \in E$ such that the total edge weight of an optimal solution to* MDBCS$_d$ *in $G^2_{u,v}$ is at least $\ell^2 - \ell$, and*

(b) *for any edge $\{u, v\} \in E$, given a solution of weight t in $G^2_{u,v}$, one can obtain in polynomial time a solution of weight $\sqrt{t}$ in G.*

Proof: We may assume, by subdividing edges, that all the edges of G have unit weight. Let $S = (V, E')$ be a connected subgraph of G of degree at most d having ℓ edges. By Remark 5.1, we can assume that S contains a cycle C. Let $\{u, v\}$ be an edge in C. Then the removal of $\{u, v\}$ does not disconnect S. Consider the subgraph H of $G^2_{u,v}$ containing ℓ copies of S, one for each edge in S, plus the corresponding contact vertices to connect the copies. Let H' be the subgraph obtained from H by removing the edge $\{u, v\}$ in each copy of S. This subgraph H' is connected by the choice of $\{u, v\}$, has maximum degree at most d, and has total weight $\ell^2 - \ell + 2\varepsilon\ell \geq \ell^2 - \ell$, proving (a).

Consider any solution H in $G^2_{u,v}$ with t edges which we want to use in order to find a solution S in G. We can ignore the edges of weight ε. Observe that subgraph H can pass from one copy of G corresponding to an edge to another copy of G corresponding to another edge only via contact vertices. To define S we distinguish two cases:

- **Case a:** H intersects at least $\sqrt{t}$ copies of G in $G^2_{u,v}$.
  Then let S be the subgraph of G induced by the edges corresponding to these copies of G in $G^2_{u,v}$.

- **Case b:** H intersects strictly fewer than $\sqrt{t}$ copies of G.
  Then let S_1 be $H \cap G_e$, with G_e being the copy of G in $G^2_{u,v}$ such that $|E(H \cap G_e)|$ is maximized. Let $e = \{x, y\}$. If S_1 is connected, we set $S = S_1$. Otherwise, S_1 has exactly two connected components C_1 and C_2, containing u and v, respectively. Since H is connected and S_1 is disconnected, necessarily the edges $\{x, u\}$ and $\{y, v\}$ belong to H. Therefore, as $\Delta_H \leq d$, the vertices u and v have degree at most $d-1$ in S_1. Let S be the graph obtained from S_1 by adding the edge $\{u, v\}$. (Recall that $G^2_{u,v}$ is defined only for $\{u, v\}$ being an edge of G.)

  In both cases S is connected, has maximum degree at most d, and has at least $\sqrt{t}$ edges. Since at least one of the cases must occur, it follows that given a solution of size t in $G^2_{u,v}$, one can obtain in polynomial time a solution of size $\sqrt{t}$ in G.

Part (b) of the lemma follows. □

Using Lemma 5.1 one can show the following lemma, similar to [156, Theorem 8].

Theorem 5.2 *For any $d \geq 2$, if* MDBCS$_d$ *has a polynomial time algorithm that achieves a constant factor approximation, then it has a* PTAS.

Proof: Again, we prove the result for $d \geq 3$, as the result for $d = 2$ follows from [156]. Let $\mathcal{A}$ be an algorithm that achieves a C-approximation for MDBCS$_d$, for a constant $C > 1$. Given the input graph $G = (V, E)$, we build the graphs $G^2_{u,v}$ for each edge $\{u, v\} \in E$, and inductively iterate this construction p times, where p is an integer to be specified later. Denote by $\mathcal{G}^{2^k}$ the set of all graphs obtained from G after k iterations.

Let OPT be the weight of an optimal solution to MDBCS$_d$ in G, and let OPT_{2^k} be the maximum weight over all graphs H in $\mathcal{G}^{2^k}$ of an optimal solution to MDBCS$_d$ in H. By Lemma 5.1(a),

$$OPT_{2^p} \geq OPT^2_{2^{p-1}} - OPT_{2^{p-1}} = OPT^2_{2^{p-1}} - o(OPT^2_{2^{p-1}}) \geq \ldots \geq OPT^{2^p} - o(OPT^{2^p}). \quad (5.5)$$

The PTAS is now obtained as follows. We run algorithm $\mathcal{A}$ on each graph $H \in \mathcal{G}^{2^p}$, and then pick the best solution among them. This yields a weight at least OPT_{2^p}/C, since $\mathcal{A}$ is a C-approximation algorithm. Beginning from this solution, we apply Lemma 5.1(b) p times to obtain a solution to MDBCS$_d$ in G with weight SOL, such that

$$SOL \geq \left(\frac{OPT_{2^p}}{C}\right)^{1/2^p} \geq \frac{\left(OPT^{2^p} - o(OPT^{2^p})\right)^{1/2^p}}{C^{1/2^p}} = \frac{OPT}{C^{1/2^p}} - o(OPT), \quad (5.6)$$

where we have used Equation (5.5). It is then clear from Equation (5.6) that for any $\varepsilon > 0$, there exists an integer $p(\varepsilon)$ such that for any graph G with $n = |V(G)|$ large enough, $\frac{OPT}{SOL} \leq 1 + \varepsilon$. Since for any fixed $\varepsilon > 0$, the number of graphs in $\mathcal{G}^{2^{p(\varepsilon)}}$ is polynomial in n, we have constructed a polynomial time $(1 + \varepsilon)$-approximation algorithm for any $\varepsilon > 0$. In other words, MDBCS$_d$ admits a PTAS for any $d \geq 3$, as claimed. □

Theorems 5.1 and 5.2 together yield the following.

Theorem 5.3 *The* MDBCS$_d$ *problem, for* $d \geq 2$*, does not admit any constant-factor approximation, unless* P = NP*.*

Karger *et al.* [156] also rule out an existence of weaker approximation for finding a longest path in a given graph. In the same spirit we show the following theorem.

Theorem 5.4 *If there is a polynomial time algorithm for* MDBCS$_d$*,* $d \geq 2$*, with performance ratio* $2^{O(\sqrt{\log n})}$*, then* NP $\subseteq$ DTIME$(2^{O(\log^5 n)})$*.*

Proof: Let $\mathcal{A}$ be an algorithm of approximation ratio $2^{O(\sqrt{\log n})}$ for MDBCS$_d$. Let $G = (V, E)$ be an instance to MDBCS$_d$ with n vertices and having an optimum solution containing ℓ edges. We choose p to be the smallest integer such that $N = n^{3^p} \geq 2^{\log^5 n}$. Let $g(n) = 2^{O(\sqrt{\log n})}$.

Now we generate the set of graphs $\mathcal{G}^{2^p}$ (defined in the proof of Theorem 5.2) by applying p times the edge squaring operation starting with G and then repeatedly applying it on previously obtained graphs. Note that the number of vertices of the elements in $\mathcal{G}^{2^p}$ is bounded above by N. By Lemma 5.1(a) we know that $\mathcal{G}^{2^p}$ contains a graph which has a solution of size at least $\ell^{2^p} - o(\ell^{2^p})$. Running $\mathcal{A}$ on all the elements of $\mathcal{G}^{2^p}$ and picking the graph with the largest number of edges, we obtain a solution H of size at least $\left(\ell^{2^p} - o(\ell^{2^p})\right)/g(N)$. Since $p = O(\log\log n)$, the number of graphs in $\mathcal{G}^{2^p}$ is bounded above by $n^{2^p} = n^{O(\log n)} = 2^{O(\log^2 n)}$. Furthermore, starting with the solution H and repeatedly applying Lemma 5.1(b), we obtain a solution to MDBCS$_d$ in G of size at least $(\ell - o(\ell))/h(n)$, where

$$h(n) = g(N)^{1/2^p} = 2^{O(\sqrt{\log N}/2^p)} = 2^{O(\log^{2.5} n/2^p)} = O(1) .$$

The last equality follows because $2^p \geq \log^{2.5} n$. Since $p = O(\log\log n)$ it follows that we have a constant factor approximation algorithm for $\ell = \Omega(\log n/\log\log n)$. If $\ell \leq c\log n/\log\log n$, for a fixed constant c, then the problem can be solved exactly in time $n^{O(\log n/\log\log n)} = 2^{O((\log n)^2/\log\log n)} \leq 2^{O(\log^5 n)}$. To find an exact solution, proceed as follows. Enumerate all subgraphs on $2c\log n/\log\log n$ vertices and at most $c\log n/\log\log n$ edges. Notice that the number of such subgraphs is upper bounded by $2^{O(\log n\log\log n)}$. Let H be one of the enumerated graphs. Then we check whether its maximum degree is upper bounded by d and whether it is connected. If both requirements are met then we check whether there is a subgraph isomorphic to H in G. To do so we use the result of Alon *et al.* [35], which shows that if H has treewidth t and $|V(H)| = h$ then one can check whether there is a subgraph isomorphic to H in G in $2^{O(h)}n^{t+1}\log n$ time. Observe that the treewidth of H is upper bounded by $\ell \leq c\log n/\log\log n$. Hence we can find an optimum solution in time

$$2^{O(\log n\log\log n)} \cdot 2^{O(\log n/\log\log n)} n^{(\log n/\log\log n)+1}\log n \leq 2^{O(\log^2 n)}.$$

This implies that we can approximate MDBCS$_d$ within a constant factor in time $2^{O(\log^5 n)}$, which is polynomial in N. But by Theorem 5.3 we know that finding a constant factor approximation to MDBCS$_d$ is NP-hard, hence we have given a simulation of a NP-hard problem in time $2^{O(\log^5 n)}$. The theorem follows. □

5.3 Approximating MDBCS$_d$

In this section we focus on approximating MDBCS$_d$. As seen in Section 5.2, MDBCS$_d$ does not admit any constant-factor approximation in general graphs.

First, we provide in Section 5.3.1 approximation algorithms in general graphs for both the weighted and unweighted versions of the problem. Then, we show in Section 5.3.2 that when the input graph has a low-degree spanning tree (in terms of d), the problem becomes easy to approximate in unweighted graphs. Specifically, Proposition 5.2 provides a constant-factor approximation for such graphs.

5.3.1 General graphs

In this section we deal with general graphs. Concerning the Longest Path problem (which corresponds to the case $d = 2$ of MDBCS$_d$ as discussed in the introduction) the best currently known approximation algorithm [57] has approximation ratio $O\left(n(\frac{\log\log n}{\log n})^2\right)$ (which was an improvement on the approximation ratio $O(n/\log n)$ of the algorithm in [35]). Using the results of [35], we provide in Theorem 5.6 an approximation algorithm for MDBCS$_d$ in general unweighted graphs for any $d \geq 2$. We then turn to weighted graphs and provide a new approximation algorithm for general weighted graphs in Theorem 5.7. Finally we compare both algorithms for unweighted graphs.

We need a preliminary lemma, that uses the following result.

Proposition 5.1 ([173]) *Any unordered tree on n nodes can be represented using $2n+o(n)$ bits with adjacency being supported in $O(n)$ time.*

Let $\mathcal{T}_{n,d}$ be the set of non-isomorphic unlabeled trees on n nodes with maximum degree at most d.

Lemma 5.2 *The set $\mathcal{T}_{\log n,d}$ can be generated in polynomial time on n.*

Proof: It is well known that $|\mathcal{T}_{n,n-1}| \sim C\alpha^n n^{-5/2}$ as $n \to \infty$, for positive constants C and α, c.f. [182]. Hence, the set $\mathcal{T}_{\log n,\log n-1}$ is of size polynomial in n. In addition, one can efficiently generate all the elements of $\mathcal{T}_{\log n,\log n-1}$. Indeed by Proposition 5.1 any unlabeled tree on $\log n$ nodes can be represented using $2\log n + o(\log n)$ bits with adjacency being supported in $O(\log n)$ time. Finally, the set $\mathcal{T}_{\log n,d}$ is obtained from $\mathcal{T}_{\log n,\log n-1}$ by removing all the elements T with $\Delta_T > d$, where Δ_T is the maximum degree of the tree T. □

The main ingredient of the first algorithm is a powerful result of [35], which uses the *color-coding* method.

Theorem 5.5 ([35]) *If a graph $G = (V, E)$ contains a subgraph isomorphic to a graph $H = (V_H, E_H)$ whose treewidth is at most t, then such a subgraph can be found in $2^{O(|V_H|)} \cdot |V|^{t+1} \cdot \log|V|$ time.*

In particular, trees on $\log|V|$ vertices can be found in time $|V|^{O(1)} \cdot \log|V|$. We are ready to describe our algorithm for unweighted graphs.

Algorithm $\mathcal{A}$:

(1) Generate all the elements of $\mathcal{T}_{\log n,d}$. Define the set $\mathcal{F} \leftarrow \emptyset$.

(2) For each $T \in \mathcal{T}_{\log n,d}$, test if G contains a subgraph isomorphic to T. If such a subgraph is found, add it to $\mathcal{F}$.

(3) If $\mathcal{F} = \emptyset$ OR $d > \log n$, output an arbitrary connected subgraph of G with d edges. Otherwise, output any element in $\mathcal{F}$.

Theorem 5.6 *For all $d \geq 2$, algorithm $\mathcal{A}$ provides a ρ-approximation algorithm for* MDBCS$_d$ *in unweighted graphs, with $\rho = \min\{m, nd/2\}/\log n$.*

Proof: Let us first observe that the running time of algorithm $\mathcal{A}$ is polynomial in n. Indeed, steps **(1)** and **(2)** can be executed in polynomial time by Lemma 5.2 and Theorem 5.5, respectively. Step **(3)** takes constant time. Algorithm $\mathcal{A}$ is clearly correct, since by definition of the set $\mathcal{T}_{\log n,d}$ the output graph is a solution of MDBCS$_d$ in G.

Finally, let us consider the approximation ratio of algorithm $\mathcal{A}$. Let OPT be the number of edges of an optimal solution of MDBCS$_d$ in G, and let ALG be the number of edges of the solution found by algorithm $\mathcal{A}$. We distinguish two cases:

- If $OPT \geq \frac{d \cdot \log n}{2}$, then any optimal solution $\hat{H}$ has at least $\log n$ vertices. In particular, $\hat{H}$ contains a tree on $\log n$ vertices, and so does G. Hence, this tree will be found in step **(2)**, and therefore $ALG \geq \log n - 1$. (We can assume that $ALG \geq \log n$ by replacing everywhere $\mathcal{T}_{\log n,d}$ with $\mathcal{T}_{\log n+1,d}$.) On the other hand, we know that $OPT \leq \min\{m, nd/2\}$.
- Otherwise, if $OPT < \frac{d \cdot \log n}{2}$, then $ALG \geq d$. Note that such a connected subgraph with d edges can be greedily found starting from any node of G.

In both cases,

$$\frac{OPT}{ALG} \leq \max\left\{\frac{\min\left\{m, \frac{nd}{2}\right\}}{\log n}, \frac{\log n}{2}\right\} = \frac{\min\{m, nd/2\}}{\log n}$$

(since $\log n = O(\sqrt{n})$), as claimed. □

In particular, if $d = 2$, Algorithm $\mathcal{A}$ reduces to the LONGEST PATH algorithm of [35].

Theorem 5.7 *The* MDBCS$_d$ *problem admits a ρ-approximation algorithm $\mathcal{B}$ in weighted graphs, with $\rho = \min\{n/2, m/d\}$.*

Proof: Let us describe the algorithm $\mathcal{B}$. Let F be the set of d heaviest edges in the input graph G, and let W be the set of endpoints of those edges. We distinguish two cases according to the connectivity of the subgraph $H = (W, F)$. Let $\omega(F)$ denote the total weight of the edges in F.

If H is connected, the algorithm returns H. We claim that this yields a ρ-approximation. Indeed, if an optimal solution consists of m^* edges of total weight ω^*, then $ALG = \omega(F) \geq \frac{\omega^*}{m^*} \cdot d$, since by the choice of F the average weight of the edges in F cannot be smaller than the average weight of the edges of an optimal solution. As $m^* \leq m$ and $m^* \leq dn/2$, we get that $ALG \geq \frac{\omega^*}{m} \cdot d$ and $ALG \geq \frac{\omega^*}{dn/2} \cdot d = \frac{\omega^*}{n/2}$.

Now suppose $H = (W, F)$ consists of a collection $\mathcal{F}$ of k connected components. Then we *glue* these components together in $k-1$ phases. In each phase, we pick two components $C, C' \in \mathcal{F}$, and combine them into a new connected component $\hat{C}$ by adding a connecting path, without touching any other connected component of $\mathcal{F}$. We then set $\mathcal{F} \leftarrow \mathcal{F} \setminus \{C, C'\} \cup \{\hat{C}\}$.

Each phase operates as follows. For every two components $C, C' \in \mathcal{F}$, compute their distance, defined as $d(C, C') = \min\{dist(u, u', G) \mid u \in C, u' \in C'\}$. Take a pair $C, C' \in \mathcal{F}$ attaining the smallest distance $d(C, C')$. Let $u \in C$ and $u' \in C'$ be two vertices realizing this distance, i.e., such that $dist(u, u', G) = d(C, C')$. Let $p(u, u')$ be a shortest path between u and u' in G. Let $\hat{C}$ be the connected component obtained by merging C, C' and the path $p(u, u')$.

For the correctness proof, we need the following two observations: First, observe that in every phase, the path $p(u, u')$ used to merge the components C and C' does not go through any other cluster C'', since otherwise, $d(C, C'')$ would be strictly smaller than $d(C, C')$, contradicting the choice of the pair (C, C'). Moreover, $p(u, u')$ does not go through any other vertex v in the cluster C except for its endpoint u, since otherwise, $dist(v, u', G) < dist(u, u', G)$, contradicting the choice of the pair u, u'. Similarly, $p(u, u')$ does not go through any other vertex v' in C'.

We now claim that after i phases, the maximum degree of H satisfies $\Delta_H \leq d - k + i + 1$. This is proved by induction on i. For $i = 0$, i.e., for the initial graph $H = (W, F)$, we observe that as F consists of d edges arranged in k separate components, the largest component will have no more than $d - k + 1$ edges, hence $\Delta_H \leq d - k + 1$, as required. Now suppose the claim holds after $i - 1$ phases, and consider phase i. All nodes other than those of the path $p(u, u')$ maintain their degree from the previous phase. The nodes u and u' increase their degree by 1, so by the inductive hypothesis, their new degree is at most $(d - k + (i - 1) + 1) + 1 = d - k + i + 1$, as required. Finally, the intermediate nodes of $p(u, u')$ have degree $2 \leq d - k + i + 1$ (since $i \geq 1$ and $k \leq d$).

It follows that by the end of phase $k - 1$, $\Delta_H \leq d - k + k - 1 + 1 = d$. Also, at that point H is connected. Hence H is a valid solution.

Finally, the approximation ratio of the algorithm is still at most $\rho = \min\{n/2, m/d\}$, since this ratio was guaranteed for the originally selected F, and the final subgraph contains the set F. □

For unweighted graphs, comparing approximation ratios of Algorithm $\mathcal{A}$ of Theorem 5.6 and Algorithm $\mathcal{B}$ of Theorem 5.7, we conclude that Algorithm $\mathcal{A}$ performs better when $d < 2 \log n$, while Algorithm $\mathcal{B}$ is better when $d \geq 2 \log n$. So if we run both the algorithms and select the best solution, we obtain the following.

Corollary 5.1 *In unweighted graphs, the* MDBCS$_d$ *problem admits a ρ-approximation algorithm, with $\rho = \min\{n/2,\ nd/(2 \log n),\ m/d,\ m/\log n\}$.*

5.3.2 Graphs with low-degree spanning trees

We first state a simple lemma about the optimal solutions of the (polynomially solvable) problem MDBS_d (the definition is the same as the MDBCS_d problem, except that the connectivity of the output subgraph is not required).

Lemma 5.3 *Given a graph G and two integers d, k , $1 < k \leq d$, such that k divides d, let OPT_d and $OPT_{d/k}$ be the optimal solutions of* MDBS_d *and* $\text{MDBS}_{d/k}$ *in G, respectively. Then $OPT_d \leq \frac{3k}{2} \cdot OPT_{d/k}$.*

Proof: Let $\hat{H}_d$ be the subgraph of G attaining OPT_d. By the classical Vizing's theorem [97], there exists a coloring of the edges of $\hat{H}_d$ using at most $d+1$ colors. Order these chromatic classes according to non-increasing total edge-weight, and let $H_{d/k}$ be the subgraph of G induced by the first d/k classes. Then the maximum degree of $H_{d/k}$ does not exceed d/k, and the sum of its edge weights is at least $\frac{d \cdot OPT_d}{k \cdot (d+1)}$. Hence

$$OPT_d \leq \frac{d+1}{d} \cdot k \cdot OPT_{d/k} \ .$$

For $d \geq 2$, the function $\frac{d+1}{d}$ is maximized when $d = 2$. □

For example, if $G = C_5$ and $d = k = 2$, then $OPT_2 = 5 \leq 3/2 \cdot 2 \cdot OPT_1 = 3 \cdot 2 = 6$.

Definition 5.2 (k-tree) *A* k-tree *of a connected graph is a spanning tree with maximum degree at most k.*

We are now ready to describe our approximation algorithm.

Proposition 5.2 *Given two integers d, ℓ , $1 < \ell < d$, let $\mathcal{G}_{d,\ell}$ be the class of graphs that have a $(d/\ell - 1)$-tree. Then, for any $G \in \mathcal{G}_{d,\ell}$,* MDBCS_d *can be approximated in G within a constant factor $\frac{3}{2}\frac{\ell}{\ell-1}$.*

Proof: Assume without loss of generality that ℓ divides d (otherwise, replace d/ℓ with $\lceil d/\ell \rceil$). Since G has a $(d/\ell - 1)$-tree, by [130], one can find in polynomial time a spanning tree S of G with maximum degree at most d/ℓ. Let $k = \frac{\ell}{\ell-1}$, and let H be the optimal solution of $\text{MDBS}_{d/k}$ in G (recall that MDBS_d is in P, but the output graph is not necessarily connected). Then it is clear that the graph $S \cup H$ is a solution of MDBCS_d in G, since it is connected and has maximum degree at most d. By Lemma 5.3 and using the fact that any solution for MDBCS_d is also a solution for MDBS_d, we conclude that $S \cup H$ provides a $\frac{3}{2}\frac{\ell}{\ell-1}$-approximation for MDBCS_d in G. □

For example, if G has a spanning tree of maximum degree at most $d/2 - 1$, then Proposition 5.2 states that MDBCS_d admits a 3-approximation in G.

The relation between MDBCS$_d$ and graph toughness

Given a graph G, denote by $\kappa(G)$ the number of connected components of G.

Definition 5.3 (Toughness of a graph [206]) *The* toughness $t(G)$ *of a graph* $G = (V, E)$ *is the largest number* t *such that, for any subset* $S \subseteq V$, $|S| \geq t \cdot \kappa(G[V \setminus S])$, *provided that* $\kappa(G[V \setminus S]) > 1$.

It is proved in [206] that if $t(G) \geq \frac{1}{k-2}$, for $k \geq 3$, then G has a k-tree.

Theorem 5.8 ([206]) *Let* G *be a graph. If* $t(G) \geq \frac{1}{k-2}$, *with* $k \geq 3$, *then* G *has a* k*-tree.*

Let us relate the above definitions with the MDBCS$_d$ problem. If a graph G does not satisfy the conditions of Proposition 5.2, then G does not have a $(d/2-1)$-tree. In this case one has some additional knowledge about the structure of G. Namely, Theorem 5.8 states that, provided that $d \geq 8$, the *toughness* $t(G)$ of G satisfies $t(G) < \frac{1}{d/2-3}$, implying that there exists a subset $S \subseteq V(G)$ such that

$$\kappa(G[V \setminus S]) > |S| \cdot \left(\frac{d}{2} - 3\right).$$

It would be interesting to explore the question whether this structural result permits to approximate MDBCS$_d$ in G efficiently.

5.4 Hardness of Approximating MSMD$_d$

The main result of this section, Theorem 5.12, states that MSMD$_d$ does not admit a constant-factor approximation on general graphs, for any fixed $d \geq 3$. We first prove in Section 5.4.1 that MSMD$_d$ does not admit a PTAS, and then use the error amplification technique to prove the main result. Our reduction is obtained from the Vertex Cover (VC) problem (see Section I.2.1).

5.4.1 MSMD$_d$ does not admit a PTAS for any $d \geq 3$

The result is first established for the case $d = 3$, in Theorem 5.9, and then extended to any $d \geq 3$ in Theorem 5.10.

Theorem 5.9 *The* MSMD$_3$ *problem does not admit a* PTAS, *unless* P = NP.

Proof: We present a gap-preserving reduction from Vertex Cover, which does not admit a PTAS in cubic graphs, unless P = NP [31]. Given a cubic graph H as instance of Vertex Cover, with $|V(H)| = n$, we construct an instance $G = f(H)$ of MSMD$_3$ as follows. Without loss of generality, we may assume that $|E(H)| = 3n/2 = 3 \cdot 2^\ell$ for some integer ℓ. Let T be the complete ternary rooted tree with root r and height $\ell+1$, which has $3 \cdot 2^\ell$ leaves and $3 \cdot 2^{\ell+1} - 2$ vertices overall. We identify the leaves of T with the elements in $E(H)$, and denote –with slight abuse of notation– this set by E (note that $E \subseteq V(T)$). We

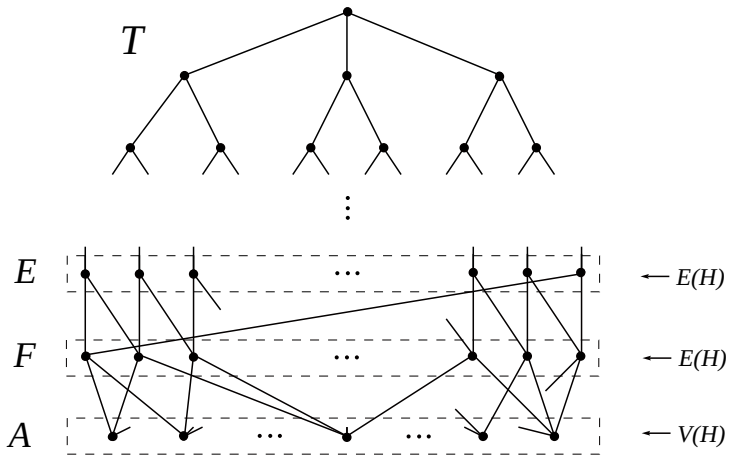

Figure 5.1: An example of the graph G built in the reduction of Theorem 5.9.

add another copy of E, called F, and a Hamiltonian cycle on $E \cup F$ inducing a bipartite graph with partition classes E and F, as shown in Fig. 5.1. We also identify the vertices of F with the elements in $E(H)$. Now we add a set A of $|V(H)|$ new vertices identified with the elements in $V(H)$, and join them to the vertices in F according to the incidence relations in H: we add an edge between a vertex in F corresponding to $e \in E(H)$ and a vertex in A corresponding to $u \in V(H)$ if and only if e contains u. This completes the construction of G, which is illustrated in Fig. 5.1.

We now claim that minimum subgraphs of G of minimum degree at least 3 correspond to minimum vertex covers of H, and vice-versa. To see this, first note that if such a subgraph D of G contains a vertex of $V(T) \cup F$, then it should contain all the vertices of $V(T) \cup F$, because of the construction of G and the degree constraints. On the other hand, D cannot contain only vertices of A (as they induce an independent set), hence D must contain all the vertices of $V(T) \cup F$. Note that all the vertices of F have degree two in $G[V(T) \cup F]$. Therefore, the problem reduces to finding a smallest subset of vertices in A covering all the vertices in F. This is exactly the Vertex Cover problem in H. Thus, we have that

$$OPT_{\mathrm{MSMD}_3}(G) \;=\; OPT_{\mathrm{VC}}(H) + |V(T)| + |F| \;=\; OPT_{\mathrm{VC}}(H) + 9n/2 - 2\ . \tag{5.7}$$

(We will omit in the sequel the reference to G and H in OPT_{MSMD_3} and OPT_{VC}, respectively.) Note also that any solution to MSMD_3 in G of size SOL_{MSMD_3} defines a solution to Vertex Cover in H of size $SOL_{\mathrm{VC}} = SOL_{\mathrm{MSMD}_3} - 9n/2 + 2$. Assume now for contradiction that MSMD_3 admits a PTAS, that is, for any $\varepsilon > 0$ we can find in polynomial time a solution to MSMD_3 in G of size $SOL_{\mathrm{MSMD}_3} \leq (1+\varepsilon) \cdot OPT_{\mathrm{MSMD}_3}$. Therefore, we could find in polynomial time a solution to Vertex Cover in H of size

$$SOL_{\mathrm{VC}} \;=\; SOL_{\mathrm{MSMD}_3} - 9n/2 + 2 \;\leq\; (1+\varepsilon) \cdot OPT_{\mathrm{MSMD}_3} - 9n/2 + 2\ . \tag{5.8}$$

Using Equation (5.7) in Equation (5.8) we get

$$SOL_{\mathrm{VC}} \;\leq\; (1+\varepsilon) \cdot OPT_{\mathrm{VC}} + \varepsilon \cdot (9n/2 - 2)\ . \tag{5.9}$$

Note that since H is cubic, any vertex cover of H has size at least $|E(H)|/3 = n/2$, so in particular $n/2 \leq OPT_{\mathrm{VC}}$. Using this inequality in Equation (5.9) yields

$$SOL_{\mathrm{VC}} \;\leq\; (1+\varepsilon)\cdot OPT_{\mathrm{VC}} + \varepsilon\cdot(9\cdot OPT_{\mathrm{VC}} - 2) \;\leq\; (1+10\varepsilon)\cdot OPT_{\mathrm{VC}}\ .$$

That is, the existence of a PTAS for MSMD_3 would imply the existence of a PTAS for Vertex Cover in cubic graphs, which is impossible unless P = NP [31]. □

Theorem 5.10 *The* MSMD_d *problem does not admit a* PTAS *for any fixed* $d \geq 3$*, unless* P = NP.

Proof: The proof consists in a generalization of the reduction presented in Theorem 5.9 for $d = 3$. Let $d \geq 3$ be a fixed integer. We present a reduction from Vertex Cover, which does not admit a PTAS in d-regular graphs, unless P = NP [31, 177]. Given a d-regular graph H as instance of Vertex Cover, with $|V(H)| = n$, we construct an instance $G = f(H)$ of MSMD_d as follows. Without loss of generality, we may assume that $|E(H)| = nd/2 = d\cdot(d-1)^\ell$, for some integer ℓ. Let T be the complete d-ary rooted tree with root r and height $\ell+1$, which has $d\cdot(d-1)^\ell$ leaves and $1+d\cdot\frac{(d-1)^{\ell+1}-1}{d-2}$ vertices overall. We identify the leaves of T with the elements in $E(H)$, and denote –with slight abuse of notation– this set by E (note that $E \subseteq V(T)$). We add another copy of E, called F, and the following edges (assuming that ℓ is big enough) according to the parity of d:

- if $d \geq 3$ is odd: $\frac{d-1}{2}$ Hamiltonian cycles on $E \cup F$, each inducing a bipartite graph with partition classes E and F.
- if $d \geq 4$ is even: $\frac{d-2}{2}$ Hamiltonian cycles on $E \cup F$, each inducing a bipartite graph with partition classes E and F, plus one perfect matching between E and F.

We also identify the vertices of F with the elements in $E(H)$. Now we add a set A of $|V(H)|$ new vertices identified with the elements in $V(H)$, and join them to the vertices in F according to the incidence relations in H: we add an edge between a vertex in F corresponding to $e \in E(H)$ and a vertex in A corresponding to $u \in V(H)$ if and only if e contains u. This completes the construction of G. Note that the vertices in E have regular degree d, and those in F have regular degree $d+1$.

As in the case $d = 3$, minimum subgraphs of G of minimum degree at least d correspond to minimum vertex covers of H, and vice-versa. Thus, we have that

$$OPT_{\mathrm{MSMD}_d}(G) \;=\; OPT_{\mathrm{VC}}(H) + |V(T)| + |E(H)| \;=\; OPT_{\mathrm{VC}}(H) + \frac{nd}{2}\cdot\frac{2d-3}{d-2} - \frac{2}{d-2}\ , \quad (5.10)$$

where we have used that $|V(T)| = 1 + d\cdot\frac{(d-1)^{\ell+1}-1}{d-2}$ and $|E(H)| = nd/2 = d\cdot(d-1)^\ell$. (We will omit in the sequel the reference to G and H in OPT_{MSMD_d} and OPT_{VC}, respectively.) Note also that any solution to MSMD_d in G of size SOL_{MSMD_d} defines a solution to Vertex Cover in H of size $SOL_{\mathrm{VC}} = SOL_{\mathrm{MSMD}_d} - \frac{nd}{2}\cdot\frac{2d-3}{d-2} + \frac{2}{d-2}$. Assume now for contradiction that MSMD_d admits a PTAS, that is, for any $\varepsilon > 0$ we can find in polynomial time a

solution to MSMD_d in G of size $SOL_{\text{MSMD}_d} \leq (1+\varepsilon) \cdot OPT_{\text{MSMD}_d}$. Therefore, we could find in polynomial time a solution to Vertex Cover in H of size

$$SOL_{\text{VC}} \leq (1+\varepsilon) \cdot OPT_{\text{MSMD}_d} - \frac{nd}{2} \cdot \frac{2d-3}{d-2} + \frac{2}{d-2} . \tag{5.11}$$

Using Equation (5.10) in Equation (5.11) we get

$$SOL_{\text{VC}} \leq (1+\varepsilon) \cdot OPT_{\text{VC}} + \varepsilon \cdot \left(\frac{nd}{2} \cdot \frac{2d-3}{d-2} - \frac{2}{d-2} \right) . \tag{5.12}$$

Note that since H is d-regular, any vertex cover of H has size at least $|E(H)|/d = n/2$, so in particular $n/2 \leq OPT_{\text{VC}}$. Using this inequality in Equation (5.12) yields

$$\begin{aligned} SOL_{\text{VC}} &\leq (1+\varepsilon) \cdot OPT_{\text{VC}} + \varepsilon \cdot \left(d \cdot \frac{2d-3}{d-2} \cdot OPT_{\text{VC}} \right) - \frac{2\varepsilon}{d-2} \\ &\leq \left(1 + \left(1 + d \cdot \frac{2d-3}{d-2} \right) \cdot \varepsilon \right) \cdot OPT_{\text{VC}} . \end{aligned}$$

That is, the existence of a PTAS for MSMD_d would imply the existence of a PTAS for Vertex Cover in d-regular graphs, which is impossible unless P = NP [31, 177]. □

5.4.2 MSMD_d is not in Apx for any $d \geq 3$

We are now ready to prove the main result of this section. Again, we focus on the case $d = 3$ in Theorem 5.11 and then extend the ideas for any $d \geq 3$ in Theorem 5.12.

Theorem 5.11 *The* MSMD_3 *problem does not admit any constant-factor approximation, unless* P = NP.

Proof: The proof is by appropriately applying the error amplification technique. Let $\mathcal{G}_1 = \{G\}$ be the family of graphs constructed in Theorem 5.9 (see Figure 5.1) from the instances H of vertex cover, G being a typical member of this family, and let $\alpha > 1$ be the factor of inapproximability of MSMD_3, that exists by Theorem 5.9.

We construct a sequence of families of graphs $\mathcal{G}_k$, such that MSMD_3 is hard to approximate within a factor $\theta(\alpha^k)$ in the family $\mathcal{G}_k$. This proves that MSMD_3 does not have any constant-factor approximation. In the following, G_k will denote a typical element of $\mathcal{G}_k$ constructed using the element G of $\mathcal{G}_1$. We describe the construction of G_2, and obtain the result by repeating the same construction inductively to obtain G_k. For every vertex v in G, we construct a graph G_v as follows. First, letting $d_v = deg_G(v)$, take a copy of G and choose d_v other arbitrary vertices $x_1, \ldots, x_{d_v}$ of degree three in $T \subset G$. Then, replace each of these vertices x_i with a cycle of length four, and join three of the vertices of the cycle to the three neighbors of x_i, $i = 1, \ldots, d_v$. Let G_v be the graph obtained in this way. Note that it contains exactly d_v vertices of degree two in G_v.

Now take a copy of G, and replace each vertex v with G_v. Then, join the d_v edges incident to v to the d_v vertices of degree two in G_v. This completes the construction of the graph G_2, which is illustrated in Figure 5.2.

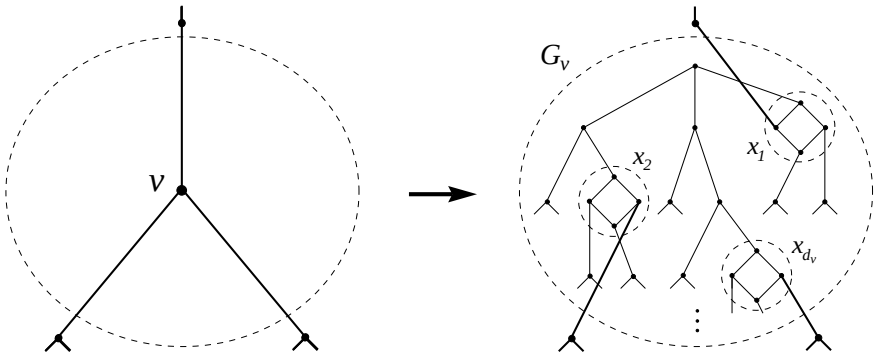

Figure 5.2: Error amplification in the proof of Theorem 5.11.

We have that $|V(G_2)| = |V(G)|^2 + o(|V(G)|^2)$, because each vertex of G is replaced with a copy of G where we had replaced some of the vertices with a cycle of length four.

To find a solution of MSMD$_3$ in G_2, note that for any $v \in V(G)$, once a vertex in G_v is chosen, we have to look for MSMD$_3$ in G, which is hard up to a constant factor α. But approximating the number of v's for which we should touch G_v is also MSMD$_3$ in G, which is hard up to the same factor α. This proves that approximating MSMD$_3$ in G_2 is hard up to a factor α^2. The proof of the theorem is completed by repeating this procedure, applying the same construction to obtain G_3, and inductively G_k. □

Theorem 5.12 *The* MSMD$_d$ *problem does not admit any constant-factor approximation for any fixed* $d \geq 3$*, unless* P = NP*.*

Proof: The proof is based on applying the error amplification technique, generalizing the proof of Theorem 5.11. Let $d \geq 3$ be a fixed integer, let $G_1 = G$ be the graph constructed in Theorem 5.10, and let $\alpha > 1$ be the factor of inapproximability of MSMD$_d$, that exists by Theorem 5.10. We construct a sequence of graphs $\mathcal{G}_k$, such that MSMD$_d$ is hard to approximate within a factor $\theta(\alpha^k)$ in $\mathcal{G}_k$. This proves that MSMD$_d$ does not have any constant-factor approximation. Indeed, suppose that MSMD$_d$ admits a C-approximation for some constant $C > 0$. Then we can choose k such that $\alpha^k > C$, and then MSMD$_d$ is hard to approximate in $\mathcal{G}_k$ within a factor $\alpha^k > C$, a contradiction.

We describe the construction of G_2, and obtain the result by repeating the same construction inductively to create G_k, a typical element of $\mathcal{G}_k$. For every vertex v in G, construct a graph G_v as follows: first, take a copy of G, and choose $d_v = deg_G(v)$ other arbitrary vertices $x_1, \ldots, x_{d_v}$ of degree d in $T \subset G$. Then, replace each of these vertices x_i with the following:

- if d is odd: a graph on $d+1$ vertices with regular degree $d-1$.
- if d is even: a graph on $d+2$ vertices having one vertex v^* of degree $d+1$, and all the others of degree $d-1$.

Next, join d of the vertices of this new graph (different from v^*) to the d neighbors of x_i, $i = 1, \ldots, d_v$. Let G_v be the graph obtained in this way. Note that we have exactly d_v vertices of degree $d-1$ in G_v.

Now, take a copy of G, and replace each vertex v with G_v. Then, join the d_v edges incident to v to the d_v vertices of degree $d-1$ in G_v. This completes the construction of the graph G_2.

We have that $|V(G_2)| = |V(G)|^2 + o(|V(G)|^2)$, because each vertex of G is replaced with a copy of G where we had replaced some of the vertices with a graph of size $d+1$ or $d+2$. The same idea of the proof of Theorem 5.11 applies to this case, proving the APX-hardness of MSMD$_d$ for $d \geq 3$. □

5.5 Approximating MSMD$_d$

In this section, it is shown that for fixed d, MSMD$_d$ is in P for graphs whose treewidth is $O(\log n)$. This is done by presenting a polynomial time algorithm based on dynamic programming. We refer to Section Section I.1.1 (page 21) for the definitions of tree decomposition and treewidth.

This dynamic programming algorithm is then used in Section 5.5.2 to provide an $O(\frac{n}{\log n})$-approximation algorithm of MSMD$_d$ for all classes of graphs excluding a fixed graph as a minor. This algorithm relies on a partitioning result for minor-excluded class of graphs, proved by Demaine et al. in [92].

5.5.1 MSMD$_d$ is in P for graphs with small treewidth

In order to prove our results, we need the following lemma which gives the time complexity of finding a smallest induced subgraph of degree at least d in graphs of bounded treewidth.

Lemma 5.4 (Lemma 6.1 of Chapter 6) *Let G be a graph on n vertices with a tree decomposition of width at most t, and let d be a positive integer. Then in time $O((d+1)^t(t+1)^{d^2}n)$, one can either find a smallest induced subgraph of minimum degree at least d in G, or identify that no such subgraph exists.*

A graph G is *q-degenerate* if every induced subgraph of G has a vertex of degree at most q. It is well known that there is a constant c such that for every h, every graph with no K_h minor is $ch\sqrt{\log h}$-degenerate [97]. This implies that M-minor-free graphs with $|M| = h$ are $ch\sqrt{\log h}$-degenerate and hence the largest value of d for which MSMD$_d$ is non-empty is $ch\sqrt{\log h}$, a constant. The above discussion, combined with the time complexity analysis mentioned in Lemma 5.4, imply the following.

Corollary 5.2 *Let G be an n-vertex graph excluding a fixed graph M as minor, with a tree decomposition of width $O(\log n)$, and let d be a positive integer constant. Then in polynomial time one can either find a smallest induced subgraph of minimum degree at least d in G, or conclude that no such subgraph exists.*

5.5.2 Approximation algorithm for M-minor-free graphs

The following result of Demaine *et al.* [92] provides a way for partitioning the vertices of a graph excluding a fixed graph as a minor into subsets with small treewidth.

Theorem 5.13 ([92]) *For a fixed graph M, there is a constant c_M such that for every integer $k \geq 1$ and for every M-minor-free graph G, the vertices of G (or the edges of G) can be partitioned into $k+1$ sets such that any k of the sets induce a graph of treewidth at most $c_M k$. Furthermore, such a partition can be found in polynomial time.*

One may assume without loss of generality that the minimum degree of the minor-free input graph $G = (V, E)$ is at least d (by removing all the vertices of lower degree), and also that $|V(G)| = n = 2^p$ for some integer $p \geq 0$ (otherwise, replace $\log n$ with $\lceil \log n \rceil$ in the description of the algorithm).

Description of the algorithm:

(1) Relying on Theorem 5.13, partition $V(G)$ in polynomial time into $\log n + 1$ sets $V_0, \ldots, V_{\log n}$ such that any $\log n$ of the sets induce a graph of treewidth at most $c_M \log n$, where c_M is a constant depending only on the excluded graph M.

(2) Run the dynamic programming algorithm of Section 5.5.1 on all the subgraphs $G_i = G[V \setminus V_i]$ of $\log n$ sets, $i = 0, \ldots, \log n$.

(3) This procedure finds all the solutions of size at most $\log n$. If no solution is found, output the whole graph G.

This algorithm clearly provides an $O(n/\log n)$-approximation for MSMD_d in minor-free graphs, for all $d \geq 3$. The running time of the algorithm is polynomial in n, since in step **(2)**, for each G_i, the dynamic programming algorithm runs in $O((d+1)^{t_i}(t_i+1)^{d^2}n)$ time, where t_i is the treewidth of G_i, which is at most $c_M \log n$.

5.6 Approximating DDDkS

We provide a deterministic approximation algorithm for the DDDkS problem in Theorem 5.14 (strongly based on the algorithm for DkS of [114]), and a randomized approximation algorithm in Theorem 5.15. Even though the performance of the randomized algorithm is worse than the one of the deterministic algorithm, we include it because the idea behind the algorithm is quite simple.

Theorem 5.14 *The* DDDkS *problem admits a deterministic $O(n^\delta)$-approximation algorithm, for some universal constant $\delta < 1/3$.*

Proof: Given an input graph G, let ρ_k^{OPT} be the optimal average degree of a subgraph of G on exactly k vertices (i.e., the optimum of DkS), and let δ_k^{OPT} be the optimal minimum degree of a subgraph of G with at most k vertices (i.e., the optimum of DDDkS). Let C be the approximation ratio of the algorithm for DkS of [114], i.e., $C = O(n^\delta)$ for some universal constant $\delta < 1/3$. Given a graph H, let ρ_H denote the average degree of H.

We know, by [114], that we can find a subgraph H_k of G on k vertices such that $\rho_{H_k} \geq \rho_k^{OPT}/C$. Removing recursively the vertices of H_k with degree strictly smaller that $\rho_{H_k}/2$ we obtain a subgraph H'_k of H_k on at most k vertices such that $\delta_{H'_k} \geq \rho_{H_k}/2 \geq \rho_k^{OPT}/(2C)$.

The next step consists of proving that there exists an integer k_0, $1 \leq k_0 \leq k$, such that $\rho_{k_0}^{OPT} \geq \delta_k^{OPT}$, so we can run the D$k$S algorithm for each $k' \leq k$, remove low-degree vertices each time, and take the best solution of DDDkS among $H'_2, H'_3, \ldots, H'_{k-1}, H'_k$.

Finally, let us prove that k_0 exists. Let H be the optimal solution of DDDkS, $\delta_H = \delta_k^{OPT}$. Let $k_0 = |V(H)|$ ($k_0 \leq k$). This is the k_0 we are looking for, because $\rho_{k_0}^{OPT} \geq \rho_H \geq \delta_H = \delta_k^{OPT}$.

The above procedure clearly constitutes a $(2C)$-approximation for DDDkS. □

Theorem 5.15 *The* DDDkS *problem admits a randomized* $O(\sqrt{n}\log n)$*-approximation algorithm.*

Proof: For every $1 \leq d \leq n$, let $H[d]$ be the maximum subgraph of G with minimum degree $\delta_{H[d]} \geq d$, in the sense that $H[d]$ contains any other subgraph H of G of minimum degree at least d. Also let $n[d] = |V(H[d])|$. The first stage of the algorithm computes $H[d]$ for every $1 \leq d \leq n$. This is easily done by initializing $H[1] = G$ and then successively removing from $H[d]$ all the vertices of degree d to obtain $H[d+1]$. Note that $n[d]$ can be zero, i.e., $H[d]$ can be the empty subgraph. The algorithm stops whenever it finds $n[d] = 0$.

Let $\tilde{d}$ be the index such that $n[\tilde{d}] > 0$ and $n[\tilde{d}+1] = 0$ (clearly $\tilde{d} \leq n-1$). If $k \geq n[\tilde{d}]$, then $H[\tilde{d}]$ is an exact solution to the problem, hence the output to the DDDkS problem is $\tilde{d}$. It remains to handle the case where $k < n[\tilde{d}]$. In this case, it is also clear that the solution d^* we are looking for is bounded by $\tilde{d}$, i.e., $d^* < \tilde{d}$. Two cases may occur.

- **Case a** : $k \leq 16\sqrt{n}\log n$ OR $\tilde{d} \leq 16\sqrt{n}\log n$.

  In this case any connected subgraph of G of size at most k (for example a connected subtree of a spanning tree of G of size k, or even just an edge) has minimum degree at least one, hence it provides a solution that is within a factor $1/(16\sqrt{n}\log n)$ of the optimal solution.

- **Case b** : Both $\tilde{d}, k > 16\sqrt{n}\log n$.

  Construct a subgraph H of $H[\tilde{d}]$ in the following way: select each vertex of $H[\tilde{d}]$ with probability $1/\sqrt{n}$, and take H to be the induced subgraph of $H[\tilde{d}]$ by the set of selected vertices. Let $n_0 = |V(H)|$.

Claim 5.2 *The number of selected vertices satisfies* $n_0 \leq 2n[\tilde{d}]/\sqrt{n}$ *with probability at least* $1 - 1/n^4$*. In particular,* $n_0 \leq k$ *with probability at least* $1 - 1/n^4$*.*

Proof: Observe that n_0 can be expressed as the sum of $n[\tilde{d}]$ independent Boolean random variables $B_1, \ldots, B_{n[\tilde{d}]}$. Since $\mathbb{E}[n_0] = n[\tilde{d}]/\sqrt{n}$, applying Chernoff's bound on the upper tail yields

$$\mathbb{P}\mathrm{rob}\left[B_1 + \cdots + B_{n[\tilde{d}]} > \frac{2n[\tilde{d}]}{\sqrt{n}}\right] < \exp\left(-\frac{n[\tilde{d}]}{4\sqrt{n}}\right).$$

Therefore, because $n[\tilde{d}] > k > 16\sqrt{n}\log n$, we have

$$\mathbb{P}\mathrm{rob}\left[n_0 > \frac{2n[\tilde{d}]}{\sqrt{n}}\right] \;<\; \exp(-4\log n) \;=\; \frac{1}{n^4}\;,$$

and since $n[\tilde{d}] \le n$, with probability at least $1 - \frac{1}{n^4}$, $n_0 \le 2n[\tilde{d}]/\sqrt{n} \le 2\sqrt{n} < 16\sqrt{n}\log n < k$. □

Claim 5.3 *For every vertex $v \in V(H)$, $deg_H(v) \ge \frac{\tilde{d}}{2\sqrt{n}}$ with probability at least $1 - 1/n^2$.*

Proof: Observe first that $deg_H(v)$ is a sum of $deg_{H[\tilde{d}]}(v)$ independent Boolean random variables, and so the expected degree of v in H is $deg_{H[\tilde{d}]}(v)/\sqrt{n} \;\ge\; \tilde{d}/\sqrt{n}$. This is because every vertex of $H[\tilde{d}]$ has degree at least $\tilde{d}$. This implies

$$\mathbb{P}\mathrm{rob}\left[deg_H(v) < \frac{\tilde{d}}{2\sqrt{n}}\right] \;\le\; \mathbb{P}\mathrm{rob}\left[deg_H(v) \;<\; \frac{deg_{H[\tilde{d}]}(v)}{2\sqrt{n}}\right].$$

Applying Chernoff's bound on the lower tail we have

$$\mathbb{P}\mathrm{rob}\left[deg_H(v) < \frac{deg_{H[\tilde{d}]}(v)}{2\sqrt{n}}\right] \;<\; \exp\left(-\frac{deg_{H[\tilde{d}]}(v)}{8\sqrt{n}}\right) \;\le\; \exp\left(-\frac{\tilde{d}}{8\sqrt{n}}\right),$$

which in turn implies (because $\tilde{d} > 16\sqrt{n}\log n$),

$$\mathbb{P}\mathrm{rob}\left[deg_H(v) < \frac{\tilde{d}}{2\sqrt{n}}\right] \;\le\; \exp\left(-\frac{16\sqrt{n}\log n}{8\sqrt{n}}\right) \;=\; \frac{1}{n^2}.$$

□

Claim 5.4 *$\delta_H \ge \tilde{d}/(2\sqrt{n})$ with probability at least $1 - 1/n$.*

Proof: By Claim 5.3, the probability that *any* node v of H has $deg_H(v) < \tilde{d}/(2\sqrt{n})$ is at most $\frac{1}{n^2} \cdot |H| \le 1/n$. □

Claim 5.2 and Claim 5.4 together show that with probability at least $1 - \frac{1}{n} - \frac{1}{n^4} \ge 1 - \frac{2}{n}$, H has at most k vertices and has minimum degree at least $\tilde{d}/(2\sqrt{n})$. Therefore, with high probability, H provides a solution of DDDkS which is within a factor $1/(2\sqrt{n})$ of the optimal solution. This concludes the proof of the theorem. □

5.7 Conclusions

This chapter considered three Degree-Constrained Subgraph problems and studied their behavior in terms of approximation algorithms and hardness of approximation. Our main results and several interesting questions that remain open are discussed below.

We proved that the MDBCS$_d$ problem is not in Apx for any $d \geq 2$, and that if there is a polynomial time algorithm for MDBCS$_d$, $d \geq 2$, with a performance ratio of $2^{O(\sqrt{\log n})}$, then NP $\subseteq$ DTIME($2^{O(\log^5 n)}$). We provided a deterministic approximation algorithm with ratio $\min\{m/\log n,\ nd/(2\log n)\}$ (resp. $\min\{n/2,\ m/d\}$) for general unweighted (resp. weighted) graphs. Finally, we gave a constant-factor approximation when the input graph has a low-degree spanning tree. Closing the huge gap between the hardness bound and the approximation ratio of our algorithm looks like a promising research direction.

We proved that the MSMD$_d$ problem is not in Apx for any $d \geq 3$. It would be interesting to strengthen this hardness result using the power of the PCP theorem. On the positive side, we gave an $O(n/\log n)$-approximation algorithm for the class of graphs excluding a fixed graph H as a minor. Finding an approximation algorithm for MSMD$_d$ in general graphs seems to be a challenging open problem. It seems that MSMD$_d$ remains hard even for proper minor-closed classes of graphs.

We provided a $O(n^\delta)$-approximation algorithm for the DDDkS problem, for some universal constant $\delta < 1/3$. It would be interesting to provide hardness results complementing this approximation algorithm. Another avenue for further research could be to consider a mixed version between DDDkS and MSMD$_d$, that would result in a two-criteria optimization problem. Namely, given a graph G, the goal would be to maximize the minimum degree while minimizing the size of the subgraph, both parameters being subject to a lower and an upper bound, respectively.

Chapter 6

Parameterized Complexity of Finding Degree-constrained Subgraphs

In this chapter we study the parameterized complexity of problem of finding degree-constrained subgraphs, taking as the parameter the number of vertices of the desired subgraph. Namely, given two positive integers d and k, we study the problem of finding a d-regular (induced or not) subgraph with at most k vertices and the problem of finding a subgraph with at most k vertices and of minimum degree at least d.

We first show that both problems are fixed-parameter intractable in general graphs. More precisely, we prove that the first problem is $W[1]$-hard using a reduction from Multi-Color Clique. The hardness of the second problem follows from an easy extension of an already know result. We then provide explicit fixed-parameter tractable (FPT) algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach. In particular, the problems become fixed-parameter tractable in planar graphs, graphs of bounded genus, and graphs with bounded maximum degree.

Keywords: parameterized complexity, degree-constrained subgraph, $W[1]$-hardness, dynamic programming, excluded minors.

6.1 Introduction

As discussed in Section III.1 (page 123), problems of finding subgraphs with certain degree constraints are well studied both algorithmically and combinatorially, and have a number of applications in network design [C8, 114, 137, 159, 160]. In this chapter we consider two natural such problems: finding a small regular (induced or not) subgraph and finding a

small subgraph with given minimum degree. We focus on these problems in Sections 6.1.1 and 6.1.2, respectively.

6.1.1 Finding a small regular subgraph

The complexity of finding regular graphs as well as regular (induced) subgraphs has been intensively studied in the literature [61, 65, 70, 75, 134, 168, 172, 195]. One of the first problems of this kind was stated by Garey and Johnson: Cubic Subgraph, that is, the problem of deciding whether a given graph contains a 3-regular subgraph, is NP-complete [75]. More generally, the problem of deciding whether a given graph contains a d-regular subgraph for any fixed degree $d \geq 3$ is NP-complete on general graphs [70] as well as on planar graphs [195] (where in the latter case only $d = 4$ and $d = 5$ were considered, since any planar graph contains a vertex of degree at most 5). Note that this problem is clearly polynomial-time solvable for $d \leq 2$. If the regular subgraph is required to be induced, Cardoso *et al.* proved that finding a maximum cardinality d-regular induced subgraph is NP-complete for any fixed integer $d \geq 0$ [65] (for $d = 0$ and $d = 1$ the problem corresponds to Maximum Independent Set and Maximum Induced Matching, respectively).

Concerning parameterized complexity of finding regular subgraphs, Moser and Thilikos proved that the following problem is $W[1]$-hard for every fixed integer $d \geq 0$ [172]:

$\geq k$-size d-Regular Induced Subgraph
Input: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does there exist a subset $S \subseteq V$, with $\|S\| \geq k$, such that $G[S]$ is d-regular?

On the other hand, the authors proved that the following problem (which can be seen as the dual of the above one) is NP-complete but has a problem kernel of size $O(kd(k+d)^2)$ for $d \geq 1$ [172]:

$\leq k$-Almost d-Regular Graph
Input: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does there exist a subset $S \subseteq V$, with $\|S\| \leq k$, such that $G[V \setminus S]$ is d-regular?

Mathieson and Szeider studied in [168] variants and generalizations of the problem of finding a d-regular subgraph (for $d \geq 3$) in a given graph by deleting at most k vertices. In particular, they answered a question of [172], proving that the $\leq k$-Almost d-Regular Graph problem (as well as some variants) becomes $W[1]$-hard when parameterized only by k (that is, it is unlikely that there exists an algorithm to solve it in time $f(k) \cdot n^{O(1)}$, where $n = |V(G)|$ and f is a function independent of n and d).

Given two integers d and k, it is also natural to ask for the existence of an induced d-regular graph with at most k vertices. The corresponding parameterized problem is defined as follows.

≤ k-SIZE d-REGULAR INDUCED SUBGRAPH (kdRIS)
Input: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does there exist a subset $S \subseteq V$, with $|S| \leq k$, such that $G[S]$ is d-regular?

Note that the complexity of ≤ k-SIZE d-REGULAR INDUCED SUBGRAPH does not follow directly from the complexity of ≥ k-SIZE d-REGULAR INDUCED SUBGRAPH as, for instance, the approximability of the problems of finding a densest subgraph on *at least* k vertices or on *at most* k vertices are significantly different [37].

In general, a graph may not contain an induced d-regular subgraph on at most k vertices, while containing a non-induced d-regular subgraph on at most k vertices. This observation leads to the following problem:

≤ k-SIZE d-REGULAR SUBGRAPH (kdRS)
Input: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does there exist a d-regular subgraph $H \subseteq G$, with $|V(H)| \leq k$?

Observe that ≤ k-SIZE d-REGULAR SUBGRAPH could a priori be easier than its corresponding induced version, as it happens for the MAXIMUM MATCHING (which is in P) and the MAXIMUM INDUCED MATCHING (which is NP-hard) problems.

To the best of our knowledge, the two parameterized problems defined above have not been considered in the literature. We prove in Section 6.2 that both problems are $W[1]$-hard for every fixed $d \geq 3$, by reduction from MULTI-COLOR CLIQUE.

6.1.2 Finding a small subgraph with given minimum degree

For a finite, simple, and undirected graph $G = (V, E)$ and $d \in \mathbb{N}$, the d-girth $g_d(G)$ of G is the minimum order of an induced subgraph of G of minimum degree at least d. The notion of d-girth was proposed and studied by Erdős *et al.* [111, 112] and Bollobás and Brightwell [60]. It generalizes the usual girth, the length of a shortest cycle, which coincides with the 2-girth. (This is indeed true because every induced subgraph of minimum degree at least two contains a cycle.) Combinatorial bounds on the d-girth can also be found in [54, 158]. The corresponding optimization problem is exactly the MSMD$_d$ problem defined in Chapter 5 (page 131). From the parameterized complexity point of view, it is natural to introduce a parameter $k \in \mathbb{N}$ and ask for the existence of a subgraph with at most k vertices and with minimum degree at least d. The problem can be formally defined as follows.

≤ k-SIZE SUBGRAPH OF MINIMUM DEGREE ≥ d (kSMDd)
Input: A graph $G = (V, E)$ and a positive integer k.
Parameter: k.
Question: Does there exist a subset $S \subseteq V$, with $|S| \leq k$, such that $G[S]$ has minimum degree at least d?

Note that the case $d = 2$ in in P, as discussed above. The special case of $d = 4$ appears in the book of Downey and Fellows [103], where it is announced that Wareham proved that kSMD4 is $W[1]$-hard. From this result, it is easy to prove that kSMDd is $W[1]$-hard for every fixed $d \geq 4$ (see Section 6.2). The case $d = 3$ remains open. Note that in the kSMDd problem we can assume without loss of generality that we are looking for the existence of an induced subgraph, since we only require the vertices to have degree at least d.

6.1.3 Presentation of the results

We do a thorough study of the kdRS, the kdRIS, and the kSMDd problems in the realm of parameterized complexity. Some basic background of parameterized complexity can be found in Section I.2.3. Our results can be classified into two categories:

General graphs: We show in Section 6.2 that kdRS is not fixed-parameter tractable by showing it to be $W[1]$-hard for any $d \geq 3$ in general graphs. We will see that the graph constructed in our reduction implies also the $W[1]$-hardness of kdRIS. In general, parameterized reductions are quite stringent because of parameter-preserving requirements of the reduction, and require some technical care. Our reduction is based on a new methodology emerging in parameterized complexity, called *multi-color clique edge representation*. This has proved to be useful in showing various problems to be $W[1]$-hard recently [72]. We first spell out step by step the procedure to use this methodology, which can be used as a template for future purposes. Then we adapt this methodology to the reduction for the kSMDd problem. Our reduction is robust, in the sense that similar problems can be shown to be $W[1]$-hard with minor modifications. The hardness of kSMDd for $d \geq 4$ follows from an easy extension of a result of Wareham [103].

Graphs with bounded local treewidth and graphs with excluded minors: Both the kSMDd and kdRS problems can be easily defined in first-order logic, where the formula only depends on k and d, both being bounded by the parameter. Frick and Grohe [128] have shown that first-order definable properties of graph classes of bounded local treewidth can be decided in time $n^{1+1/k}$ for all k, in particular in time n^2, and first-order model checking is FPT on M-minor-free graphs. This immediately gives us the *classification result* that both problems are FPT for $d \geq 3$ in graphs with bounded local treewidth and graphs excluding a fixed graph M as a minor. These classification results can be generalized to a larger class of graphs, namely graphs locally excluding a fixed graph M as a minor, by a recent result of Dawar, Grohe and Kreutzer [89]. These results are by nature very general and can involve huge coefficients (dependence on k). A natural problem arising in this context is then the design of an explicit algorithm for kSMDd for $d \geq 3$ in these graph classes with explicit time complexity, faster than the one coming from the meta-theorem of Frick and Grohe. In Section 6.3, we provide a fast and explicit algorithms for kSMDd, $d \geq 3$, in graphs with bounded local treewidth and graphs excluding a fixed graph M as a minor. For the sake of simplicity, we present the algorithm for the kSMDd problems, but the same algorithms can be applied to the kdRS problem, with the same time bounds. Our algorithms use standard dynamic programming over graphs with bounded treewidth and

a few results concerning the clique decomposition of M-minor-free graphs developed by Robertson and Seymour in their graph minor theory [187]. A set of non-trivial observations allow to get improvements in the time complexity of the algorithms. We note that our dynamic programming over graphs with bounded treewidth is also generic and can handle variations on degree-constrained subgraph problems with simple changes.

6.2 Fixed-Parameter In-tractability Results

As mentioned in the introduction, kSMDd is known to be $W[1]$-hard for $d = 4$ [103]. It can be easily proved that kSMDd is $W[1]$-hard for every $d \geq 4$, by reducing kSMDd to $k\text{SMD}_{d+1}$.

Indeed, let G be an instance of kSMDd, with parameter k. We construct an instance G' of $k\text{SMD}_{d+1}$ from G by adding a vertex u and connecting it to all the vertices of G. We set the parameter to $k+1$. If there is a subset of vertices $S \subseteq V(G)$ of size at most k and with minimum degree at least d, then $S \cup \{u\}$ is a solution to $k\text{SMD}_{d+1}$ in G' (the degree of u is also at least $d+1$ since we can assume that $k \geq d+1$). Conversely, if there is a subset of vertices $S \subseteq V(G')$ of size at most $k+1$ and with minimum degree at least $d+1$, we construct a solution to kSMDd in G as follows.

- if $u \in S$, then $S \setminus \{u\}$ is a solution in G.
- otherwise, if $u \notin S$, let v be an arbitrary vertex in S. Then any connected component of the subgraph induced by $S \setminus \{v\}$ is a solution in G, since $|S \setminus \{v\}| \leq k$ and the degrees of the vertices in $S \setminus \{v\}$ have decreased by at most 1 after the removal of v.

In the remainder of this section we give a $W[1]$-hardness reduction for kdRS. The definition of a parameterized reduction can be found in Section I.2.3 (page 24). Our reduction is from Multi-Color Clique, which is known to be $W[1]$-complete by a simple reduction from the ordinary Clique [115], and is based on the methodology known as *multi-color edge representation*. The Multi-Color Clique problem is defined as follows.

Multi-color Clique
Input: An graph $G = (V, E)$, a positive integer k, and a proper k-coloring of $V(G)$.
Parameter: k.
Question: Does there exist a clique of size k in G consisting of exactly one vertex of each color?

Consider an instance $G = (V, E)$ of Multi-color Clique with its vertices colored with the set of colors $\{c_1, \cdots, c_k\}$. Let $V[c_i]$ denote the set of vertices of color c_i. For each edge $e = \{u, v\}$ of G, with $u \in V[c_i]$, $v \in V[c_j]$, and $i < j$, we first replace e with two arcs $e^f = (u, v)$ and $e^b = (v, u)$. By abuse of notation, we also call this digraph G. Let $E[c_i, c_j]$ be the set of arcs $e = (u, v)$, with $u \in V[c_i]$ and $v \in V[c_j]$, for $1 \leq i \neq j \leq k$. An arc $(u, v) \in E[c_i, c_j]$ is called *forward* (resp. *backward*) if $i < j$ (resp. $i > j$). We also assume that $|V[c_i]| = N$ for all i, and that $|E[c_i, c_j]| = M$ for all $i \neq j$, i.e., we assume that the color classes of G,

and also the arc sets between them, have uniform sizes. For a simple justification of this assumption, we can reduce Multi-color Clique to itself, taking the union of $k!$ disjoint copies of G, one for each permutation of the color sets.

In this methodology, the basic encoding bricks correspond to the arcs of G, which we call **arc gadgets**. We generally have three kinds of gadgets, which we call **selection**, **coherence**, and **match gadgets**. These are engineered together to get an overall reduction gadget for the problem. In an optimal solution to the problem (that is, a solution providing a Yes answer), the selection gadget ensures that *exactly one* arc gadget is selected among arc gadgets corresponding to arcs going from a color class $V[c_i]$ to another color class $V[c_j]$. For any color class $V[c_i]$, the coherence gadget ensures that the out-going arcs from $V[c_i]$, corresponding to the selected arc gadgets, have a common vertex in $V[c_i]$. That is, all the arcs corresponding to these selected arc gadgets *emanate from the same vertex in* $V[c_i]$. Finally, the match gadget ensures that if we have selected an arc gadget corresponding to an arc (u, v) from $V[c_i]$ to $V[c_j]$, then the arc gadget selected from $V[c_j]$ to $V[c_i]$ corresponds to (v, u). That is, *both of* e^f *and* e^b *are selected together.* In what follows, we show how to particularize this general strategy to obtain a reduction from Multi-color Clique to kdRS for $d \geq 3$. To simplify the presentation, we first describe our reduction for the case $d = 3$ (in Section 6.2.1) and then we describe the required modifications for the case $d \geq 4$ in Section 6.2.2.

6.2.1 $W[1]$-hardness for the cubic case

In this section we give in detail the construction of all the gadgets for $d = 3$. Recall that an arc $(u, v) \in E[c_i, c_j]$ is forward if $i < j$, and it is backward if $i > j$. We refer the reader to Figure 6.1 to get an idea of the construction.

Arc gadgets: For each arc $(u, v) \in E[c_i, c_j]$ with $i < j$ (resp. $i > j$) we have a cycle C_{e^f} (resp. C_{e^b}) of length $3 + 2(k-2) + 2$, with the set of vertices:

- *selection vertices:* e^f_{s1}, e^f_{s2}, and e^f_{s3} (resp. e^b_{s1}, e^b_{s2}, and e^b_{s3});
- *coherence vertices:* e^f_{ch1r}, e^f_{ch2r} (resp. e^b_{ch1r}, e^b_{ch2r}), for all $r \in \{1, \ldots, k\}$ and $r \neq i, j$; and
- *match vertices:* e^f_{m1} and e^f_{m2} (resp. e^b_{m1} and e^b_{m2}).

Selection gadgets: For each pair of indices i, j with $1 \leq i \neq j \leq k$, we add a new vertex A_{c_i,c_j}, and connect it to all the selection vertices of the cycles C_{e^f} if $i < j$ (resp. C_{e^b} if $i > j$) for all $e \in E[c_i, c_j]$. This gadget is called *forward selection gadget* (resp. *backward selection gadget*) if $i < j$ (resp. $i > j$), and it is denoted by $\mathcal{S}_{i,j}$.

That is, we have $k(k-1)$ clusters of gadgets: one gadget $\mathcal{S}_{i,j}$ for each set $E[c_i, c_j]$, for $1 \leq i \neq j \leq k$.

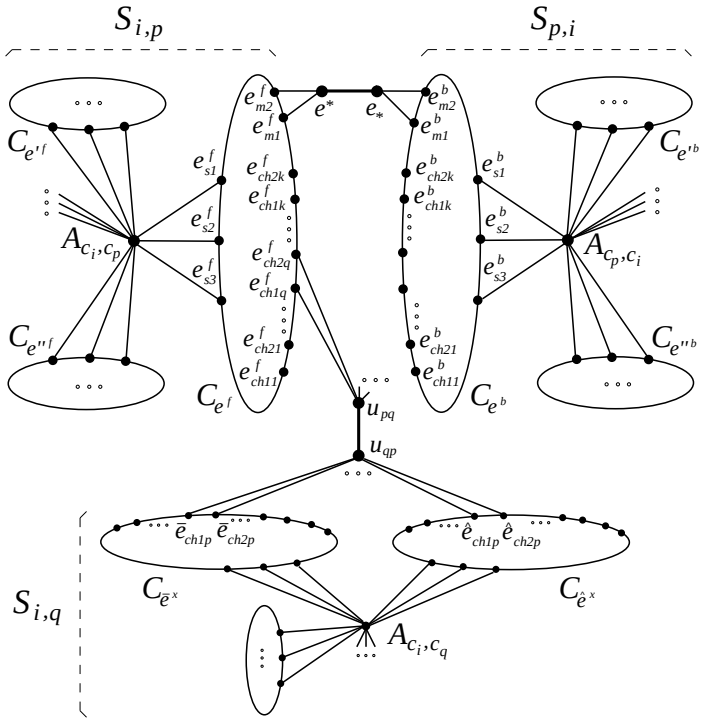

Figure 6.1: Gadgets used in the reduction of the proof of Theorem 6.1 (we suppose $i < p$).

Coherence gadgets: For each i, $1 \leq i \leq k$, let us consider all the selection gadgets of the form $\mathcal{S}_{i,p}$, $p \in \{1, \cdots, k\}$ and $p \neq i$. For any $u \in V[c_i]$, and any two indices $1 \leq p \neq q \leq k$, $p, q \neq i$, we add two new vertices u_{pq} and u_{qp}, and a new edge $\{u_{pq}, u_{qp}\}$. For every arc $e = (u, v) \in E[c_i, c_p]$, with $u \in V[c_i]$, we pick the cycle C_{e^x}, $x \in \{f, b\}$ depending on whether e is forward or backward, and add two edges of the form $\{e_{ch1q}, u_{pq}\}$ and $\{e_{ch2q}, u_{pq}\}$. Similarly, for an arc $e = (u, w) \in E[c_i, c_q]$, with $u \in V[c_i]$, we pick the cycle C_{e^x}, $x \in \{f, b\}$, and add two edges $\{e_{ch1p}, u_{qp}\}$ and $\{e_{ch2p}, u_{qp}\}$.

Match gadgets: For any pair of arcs $e^f = (u, v)$ and $e^b = (v, u)$, we consider the two cycles C_{e^f} and C_{e^b} corresponding to e^f and e^b. Now, we add two new vertices e^* and e_*, a *matching edge* $\{e^*, e_*\}$, and all the edges of the form $\{e^f_{m1}, e^*\}$, $\{e^f_{m2}, e^*\}$, $\{e^b_{m1}, e_*\}$ and $\{e^b_{m2}, e_*\}$ where e^f_{m1}, e^f_{m2} are match vertices on C_{e^f}, and e^b_{m1} , e^b_{m2} are match vertices on C_{e^b}.

This completes the construction of the gadgets, and the union of all of them defines the graph $\mathscr{G}_G$ depicted in Figure 6.1.

We now prove that this construction yields the reduction through a sequence of simple claims.

Claim 6.1 *Let G be an instance of* Multi-color Clique*, and $\mathscr{G}_G$ be the graph we constructed above. If G has a multi-colored k-clique, then $\mathscr{G}_G$ has a 3-regular subgraph of size $k' = (3k+1)k(k-1)$.*

Proof: Let ω be a multi-color clique of size k in G. For every edge $e \in E(\omega)$, select the corresponding cycles C_{e^f}, C_{e^b} in $\mathscr{G}_G$. Let us define S as follows.

$$S = \bigcup_{e\in\omega, x\in\{f,b\}} N_{\mathscr{G}_G}[V(C_{e^x})] \ .$$

It is straightforward to check that $\mathscr{G}_G[S]$ is a 3-regular subgraph of $\mathscr{G}_G$. To verify the size of $\mathscr{G}_G[S]$, note that we have $2 \cdot \binom{k}{2}$ cycles in $\mathscr{G}_G[S]$ and each of them contributes $(3k-1)$ vertices (this includes vertices on the cycle themselves). □

Claim 6.2 *Any 3-regular subgraph of $\mathscr{G}_G$ contains one of the cycles C_{e^x}, $x \in \{b, f\}$, corresponding to arc gadgets.*

Proof: Note that if such a subgraph of $\mathscr{G}_G$ intersects a cycle C_{e^x}, then it must contain all of its vertices. Further, if we remove all the vertices corresponding to arc gadgets in $\mathscr{G}_G$, then the remaining graph is a forest. These two facts together imply that any 3-regular subgraph of $\mathscr{G}(G)$ should intersect at least one cycle C_{e^x} corresponding to an arc gadget, hence it must contain C_{e^x}. □

Claim 6.3 *If $\mathscr{G}_G$ contains a 3-regular subgraph of size $k' = (3k+1)k(k-1)$, then G has a multi-colored k-clique.*

Proof: Let $H = G[S]$ be a 3-regular subgraph of size k'. Now, by Claim 6.2, S must contain all the vertices of a cycle corresponding to an arc gadget. Furthermore, notice that to ensure the degree condition in H, once we have a vertex of a cycle in S, all the vertices of this cycle and their neighbors are also in S. Without loss of generality, let C_{e^f} be this cycle, and suppose that it belongs to the gadget $\mathcal{S}_{i,j}$, i.e., $e \in E[c_i, c_j]$ and $i < j$. Notice that by construction, this forces some of the other vertices to belong also to S. Indeed, its match vertices force the cycle C_{e^b} of $\mathcal{S}_{j,i}$ to be in S. The coherence vertices of C_{e^f} force S to contain at least one cycle in $\mathcal{S}_{i,l}$, for all $l \in \{1, \cdots, k\}$, $l \neq i$. They in turn force S to contain at least one cycle from the remaining gadgets $\mathcal{S}_{p,q}$ for all $p \neq q \in \{1, \ldots, k\}$. The selection vertices of each such cycle in $\mathcal{S}_{p,q}$ force S to contain $A_{p,q}$. But because of our condition on the size of S ($|S| = k'$), we can select *exactly one* cycle gadget from each of the gadgets $\mathcal{S}_{p,q}$, $p \neq q \in \{1, 2, \cdots, k\}$. Let E' be the set of edges in $E(G)$ corresponding to arc gadgets selected in S. We claim that $G[V[E']]$ is a multi-color clique of size k in G. Here $V[E']$ is a subset of vertices of $V(G)$ containing the end points of the edges in E'. First of all, because of the match vertices, once e^f is in E', e^b is forced to be in E'. To conclude the proof we only need to ensure that all the edges from a particular color class emanate from the same vertex. But this is ensured by the restriction on the size of S and the presence of coherence vertices on the cycles selected in S from $\mathcal{S}_{p,q}$, $p \neq q \in \{1, 2, \cdots, k\}$. To see this, let us take two arcs $e = (u, v) \in (E[c_i, c_p] \cap E')$ and $e' = (u', w) \in (E[c_i, c_q] \cap E')$. Now the four vertices u_{pq}, u_{qp}, u'_{pq}, and u'_{qp} belong to S. If u is different from u', then S has at least two elements more than the expected size k', which contradicts the condition on the size of S. All these facts together imply that $G[V(E')]$ forms a multi-colored k-clique in the original graph G. □

Claims 6.1 and 6.3 together yield the following theorem:

Theorem 6.1 *k*3RS *is* W[1]*-hard.*

We shall see in the next section that the proof of the Theorem 6.1 can be generalized to larger values of d. Note that the 3-regular subgraph constructed in the proof of Theorem 6.1 is a 3-regular *induced* subgraph, so our proof implies the following corollary.

Corollary 6.1 *k*3RIS *is* W[1]*-hard.*

6.2.2 *W*[1]-hardness for higher degrees

In this section we generalize the reduction given in Section 6.2.1 for $d \geq 4$. The main idea is to change the role of the cycles C_e by $(d-1)$-regular graphs of appropriate size. We show below all the necessary changes in the construction of the gadgets to ensure that the proof for $d = 3$ works for $d \geq 4$.

Arc gadgets for $d \geq 4$***:*** Let us take C to be a $(d-1)$-regular graph of size $(d-1)+(d-1)(k-2)+d$, if it exists (that is, if $(d-1)$ is even or k is odd). If such a graph does not exist, we take a graph of size $(d-1)+(d-1)(k+2)+d+1$ and with regular degree $d-1$ on the set C of $(d-1)+(d-1)(k+2)+d$ vertices and degree d on the last vertex v. As before, we replace each edge e with two arcs e^f and e^b. For each arc $e^x \in E[c_i, c_j]$, we add a copy of C, that we call C_{e^x}, with the following vertex set:

- *selection vertices:* $e^x_{s1}, e^x_{s2}, \cdots, e^x_{sd}$;
- *coherence vertices:* $e^x_{ch1r}, \cdots, e^x_{ch(d-1)r}$, for all $r \in \{1, \ldots, k\}$, $r \neq i, j$; and
- *match vertices:* $e^x_{m1}, \cdots, e^x_{m(d-1)}$.

Selection gadgets for $d \geq 4$***:*** Without loss of generality suppose that $x = f$. As before, we add a vertex A_{c_i,c_j}, and for every arc $e^f \in E[c_i, c_j]$ we add all the edges from A_{c_i,c_j} to all the selection vertices of the graph C_{e^f}. We call this gadget $\mathcal{S}_{i,j}$.

Coherence gadgets for $d \geq 4$***:*** Fix an i, $1 \leq i \leq k$. Let us consider all the selection gadgets of the form $\mathcal{S}_{i,p}$, $p \in \{1, \cdots, k\}$ and $p \neq i$. For any $u \in V[c_i]$, and any two indices $p \neq q \leq k$, $p, q \neq i$, we add a new edge $\{u_{pq}, u_{qp}\}$. For every arc $e = (u, v) \in E[c_i, c_p]$, with $u \in V[c_i]$, we pick the graph C_{e^x}, $x \in \{f, b\}$, depending on whether e is forward or backward, and add $d-1$ edges of the form $\{e_{ch1q}, u_{pq}\}, \{e_{ch2q}, u_{pq}\}, \ldots, \{e_{ch(d-1)q}, u_{pq}\}$. Similarly, for an arc $e = (u, w) \in E[c_i, c_q]$, with $u \in V[c_i]$, we pick the graph C_{e^x}, $x \in \{f, b\}$, and add $d-1$ edges of the form $\{e_{ch1p}, u_{qp}\}, \ldots, \{e_{ch(d-1)p}, u_{qp}\}$.

Match gadgets for $d \geq 4$: For the two arcs $e^f = (u, v)$ and $e^b = (v, u)$, we consider the two graphs C_{e^f} and C_{e^b} corresponding to e^f and e^b. Now we add a matching edge $\{e^*, e_*\}$ and add all the edges of the form $\{e^f_{m1}, e^*\}, \ldots, \{e^f_{m(d-1)}, e^*\}$ and $\{e^b_{m1}, e_*\}, \ldots, \{e^b_{m1}, e_*\}$, where e^f_{mi}, e^b_{mi} are match vertices of C_{e^f} and of C_{e^b}, respectively.

This completes the construction of the gadgets, and the union of all of them defines the graph $\mathscr{G}_G$. It is not hard to see that a proof similar to that of Theorem 6.1 shows that G, an instance of multi-color clique, has a multi-colored clique of size k if and only if $\mathscr{G}_G$ has a d-regular subgraph of size $k' = dk + 1$. We have the following theorem.

Theorem 6.2 *kd*RS *is* W[1]*-hard for all $d \geq 3$.*

Notice that again the d-regular subgraph constructed in the proof of Theorem 6.2 turns out to be an induced subgraph of regular degree d in $\mathcal{G}_G$. As a consequence we obtain the following corollary:

Corollary 6.2 *kd*RIS *is* W[1]*-hard for all $d \geq 3$.*

6.3 FPT Algorithms for Graphs with Bounded Local Treewidth and Graphs with Excluded Minors

In this section, we provide explicit (and fast) algorithms for kSMDd, $d \geq 3$, in graphs with bounded local treewidth (Section 6.3.1) and in graphs excluding a fixed graph M as a minor (Section 6.3.2). We first provide the necessary background.

The definition of treewidth (see page 21) can be generalized to take into account the local properties of G, and this is called *local treewidth*. To define it formally, we first need to define the r-neighborhood of vertices of G. The *distance* $d_G(u, v)$ between two vertices u and v of G is the length of a shortest path in G from u to v. For $r \geq 1$, a *r-neighborhood* of a vertex $v \in V$ is defined as $N^r_G(v) = \{u \in V \mid d_G(v, u) \leq r\}$.

The *local treewidth* of a graph G is a function $ltw^G : \mathbb{N} \to \mathbb{N}$ which associates to every integer $r \in \mathbb{N}$ the maximum treewidth of an r-neighborhood of vertices of G, i.e.,

$$ltw^G(r) = \max_{v \in V(G)} \{tw(G[N^r_G(v)])\}.$$

A graph class $\mathscr{G}$ has *bounded local treewidth* if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that for each graph $G \in \mathscr{G}$ and for each integer $r \in \mathbb{N}$, we have $ltw^G(r) \leq f(r)$. For a given function $f : \mathbb{N} \to \mathbb{N}$, $\mathscr{G}_f$ is the class of all graphs G of local tree-width at most f, i.e., such that $ltw^G(r) \leq f(r)$ for every $r \in \mathbb{N}$. We refer to [108] and [141] for more details.

We now provide the basics to understand the structure of the classes of graphs excluding a fixed graph as a minor.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two disjoint graphs, and $k \geq 0$ an integer. For $i = 1, 2$, let $W_i \subseteq V_i$ form a clique of size h and let G'_i be the graph obtained from G_i by removing a set of edges (possibly empty) from the clique $G_i[W_i]$. Let $F : W_1 \to W_2$ be

a bijection between W_1 and W_2. The *h-clique sum* or the *h-sum* of G_1 and G_2, denoted by $G_1 \oplus_{h,F} G_2$, or simply $G_1 \oplus G_2$ if there is no confusion, is the graph obtained by taking the union of G_1' and G_2' by identifying $w \in W_1$ with $F(w) \in W_2$, and by removing all the multiple edges. The image of the vertices of W_1 and W_2 in $G_i \oplus G_2$ is called the *join* of the sum.

Note that $\oplus$ is not well defined; different choices of G_i' and the bijection F can give different clique sums. A sequence of h-sums, not necessarily unique, which result in a graph G, is called a *clique sum decomposition* or, simply, a *clique decomposition* of G.

Let Σ be a surface with boundary cycles $C_1, \ldots, C_h$. A graph G is *h-nearly embeddable* in Σ, if G has a subset X of vertices of size at most h, called *apices*, such that there are (possibly empty) subgraphs $G_0, \ldots, G_h$ of $G \setminus X$ such that

1. $G \setminus X = G_0 \cup \cdots \cup G_h$;
2. G_0 is embeddable in Σ (we fix an embedding of G_0);
3. $G_1, \ldots, G_h$ are pairwise disjoint;
4. For $1 \leq \cdots \leq h$, let $U_i := \{u_{i_1}, \ldots, u_{i_{m_i}}\} = V(G_0) \cap V(G_i)$, G_i has a path-decomposition $(\{B_{ij}\},\ 1 \leq j \leq m_i)$ of width at most h such that
   (a) for $1 \leq i \leq h$ and for $1 \leq j \leq m_i$ we have $u_j \in B_{ij}$; and
   (b) for $1 \leq i \leq h$, we have $V(G_0) \cap C_i = \{u_{i_1}, \ldots, u_{i_{m_i}}\}$ and the points $u_{i_1}, \ldots, u_{i_{m_i}}$ appear on C_i in this order (either walking through the cycles clockwise or counterclockwise).

6.3.1 Graphs with bounded local treewidth

In order to prove our results, we need the following lemma, which gives the time complexity of finding a smallest induced subgraph of degree at least d in graphs with bounded treewidth.

Lemma 6.1 *Let G be a graph on n vertices with a tree-decomposition of width at most t, and let d be a positive integer. Then in time $O((d+1)^t(t+1)^{d^2}n)$ we can decide whether there exists an induced subgraph of degree at least d in G and, if such a subgraph exists, find one of the smallest size.*

Proof: Let $(T, \mathcal{X})$ be the given tree-decomposition. We assume that T is a rooted tree, and that the decomposition is *nice*, which means the following:

- Each node has at most two children;
- For every node t with exactly two children t_1 and t_2, $X_t = X_{t_1} = X_{t_2}$;
- For every node t with exactly one child s, either $X_t \subset X_s$ and $|X_s| = |X_t| + 1$, or $X_s \subset X_t$ and $|X_t| = |X_s| + 1$.

Note that such a decomposition always exists and can be found in linear time, and in fact we may assume that $|V(T)| = O(n)$. As usual in algorithms based on tree decompositions,

we employ a dynamic programming approach based on this decomposition, which at the end either produces a connected subgraph of G of minimum degree at least d and of size at most k, or decides that G does not have any such subgraph.

As the tree decomposition is rooted, we can speak of the subgraph defined by the subtree rooted at node i. More precisely, for any node i of T, let Y_i be the set of all vertices that appear either in X_i or in X_j for some descendant j of i. Denote by $G[Y_i]$ the graph induced by the nodes in Y_i.

Note that if i is a node in the tree and j_1 and j_2 are two children, then Y_{j_1} and Y_{j_2} are disjoint except for vertices in X_i, i.e., $Y_{j_1} \cap Y_{j_2} = X_i$. A $\mathcal{P}$-coloring of the vertices in X_i, for the palette $\mathcal{P} = \{0, 1, \ldots, d\}$, is a function $c_i : X_i \to \mathcal{P}$. The *support* of c is $supp(c) = \{v \in X_i \mid c(v) \neq 0\}$.

For any such $\mathcal{P}$-coloring c of vertices in X_i, let $a(i, c)$ be the minimum size of an induced subgraph $H(i, c)$ of $G[Y_i]$, which has degree $c(v)$ for every $v \in X_i$ with $c(v) \neq d$, and degree at least d on its other vertices. Note that $H(i, c) \cap X_i = supp(c)$. If such a subgraph does not exist, we define $a(i, c) = +\infty$.

We develop recursive formulas for $a(i, c)$. In the base case, i is a leaf of the tree decomposition. Hence $Y_i = X_i$. The size of the minimum induced subgraph with prescribed degrees is exactly $|supp(c)|$ if $G[supp(c)]$ satisfies the degree conditions, and is $+\infty$ if it does not.

In the recursive case, node i has at least one child. We distinguish between three cases, depending on the size of the bag of i and its number of children.

Case (1): i has only one child j and $X_i \subset X_j$.
Then $|X_j| = |X_i| + 1$ and $X_i = X_j \setminus \{v\}$ for some vertex v. Also, $Y_i = Y_j$, since X_i does not add any new vertices. Consider a coloring $c : X_i \to \mathcal{P}$. Consider the two colorings $c_0 : X_j \to \mathcal{P}$ and $c_1 : X_j \to \mathcal{P}$ of X_j, defined as follows: $c_0 = c_1 = c$ on X_i, and $c_0(v) = 0$, $c_1(v) = d$. Then we let $a(i, c) = \min\{a(j, c_0), a(j, c_1)\}$.

Case (2): i has only one child j and $X_j \subset X_i$.
Then $|X_j| = |X_i| - 1$ and $X_j = X_i \setminus \{v\}$ for some vertex v. Also, $Y_j = Y_i \setminus \{v\}$. Let c be a coloring of X_i. It is clear that the only neighbors of v in $G[Y_i]$ are already in X_i.

- If $c(v) \geq 1$, for any collection $\mathcal{A}$ of $c(v)$ edges in $G[X_i]$ connecting v to vertices $v_1, \ldots, v_{c(v)}$, with $c(v_i) \geq 1$ (note that such a collection may not exist at all), we consider the coloring $c_{\mathcal{A}}$ of X_j as follows: $c_{\mathcal{A}}(v_i) = c(v_i) - 1$ for any $1 \leq i \leq c(v)$, and $c_{\mathcal{A}}(w) = c(w)$ for any other vertex w. Then we define

$$a(i, c) = \min_{\mathcal{A}}\{a(j, c_{\mathcal{A}})\} + 1 \ .$$

- If $c(v) = 0$, we simply define $a(i, c) = a(j, c)$.

Note that there are at most $(t + 1)^{d+1}$ choices for such a collection $\mathcal{A}$.

Case (3): i has two children j_1 and j_2.
Then $X_i = X_{j_1} = X_{j_2}$. Let c be a coloring of X_i, then $supp(c) \subset X_i$ is part of the subgraph we are looking for. For any vertex $v \in X_i$, calculate the degree $deg_{G[X_i]}(v)$. Suppose that v has degree d_1^v, d_2^v in $H \cap G[Y_{j_1}], H \cap G[Y_{j_2}]$ (H is the subgraph we are looking for). These

degree sequences should guarantee the degree condition on v imposed by the coloring c. In other words, if $c(v) \leq d-1$ then we should have $d_1^v + d_2^v - d_{G[X_i]} = c(v)$, and if $c(v) = d$, then $d_1^v + d_2^v - d_{G[X_i]} \geq d$. Every such sequence $\mathcal{D} = \{d_1^v, d_2^v \mid v \in X_i\}$ on vertices of X_i determines two colorings $c_1^{\mathcal{D}}$ and $c_2^{\mathcal{D}}$ of X_{j_1} and X_{j_2} respectively. For each such pair of colorings, let H_1 and H_2 be the minimum subgraphs with these degree constraints in $G[Y_{j_1}]$ and $G[Y_{j_2}]$ respectively. Then $H_1 \cup H_2$ satisfies the degree constraints imposed by c. We define

$$a(i,c) = \min_{\mathcal{D}}\{|H| \mid H = H_1 \cup H_2\}$$

for all degree distributions as above. For every vertex we have at most d^2 possible degree choices for d_1^v and d_2^v. We have also $|X_i| \leq t+1$. This implies that the minimum is taken over at most $(t+1)^{d^2}$ colorings.

As the size of our tree-decomposition is linear on n, we can determine all the values $a(i,c)$ for every $i \in V(T)$ and every coloring of X_i in time linear in n. Now return the minimum value of $a(i,c)$ computed for all colorings c, for values in the set $\{0,d\}$ assigning at least one non-zero value. The time dependence on t follows from the size of the bags and the choices made using the colorings. □

Lemma 6.1 leads to the following theorem:

Theorem 6.3 *For any $d \geq 3$ and any function $f : \mathbb{N} \to \mathbb{N}$, kSMDd is fixed-parameter tractable on $\mathscr{G}_f$. Furthermore, the algorithm runs in time $O((d+1)^{f(2k)}(f(2k)+1)^{d^2}n^2)$.*

Proof: Let $G = (V,E)$ be a graph in $\mathscr{G}_f$, that is, G has bounded local treewidth and the bound is given by the function f. We first notice that if there exists an induced subgraph $H \subseteq G$ of size at most k and degree at least d, then H can be supposed to be connected. Secondly, if we know a vertex v of H, then H is contained in $N_G^k[v]$, which has diameter at most $2k$. Hence there exists the desired H if and only if there exists $v \in V$ such that H is contained in $N_G^k[v]$. To solve the problem, for each $v \in V$, we find a tree-decomposition of $N_G^k[v]$ of width at most $f(2k)$ in time polynomial in n, and then run the algorithm of Lemma 6.1. □

The function $f(k)$ is known to be $3k$, $C_g gk$, and $b(b-1)^{k-1}$ for planar graphs, graphs of genus g, and graphs of degree at most b, respectively [108, 141]. Here C_g is a constant depending only on the genus g of the graph. As an easy corollary of Theorem 6.3, we have the following:

Corollary 6.3 *kSMDd can be solved in $O((d+1)^{6k}(6k+1)^{d^2}n^2)$, $O((d+1)^{2C_g gk}(2C_g gk+1)^{d^2}n^2)$ and $O((d+1)^{2b(b-1)^{k-1}}(2b(b-1)^{k-1}+1)^{d^2}n^2)$ time in planar graphs, graphs of genus g, and graphs of degree at most b, respectively.*

6.3.2 M-minor-free graphs

In this section, we consider the class of M-minor-free graphs. We need the following theorem of Robertson and Seymour [187] (see also Demaine et al. in [92] for an algorithmic version).

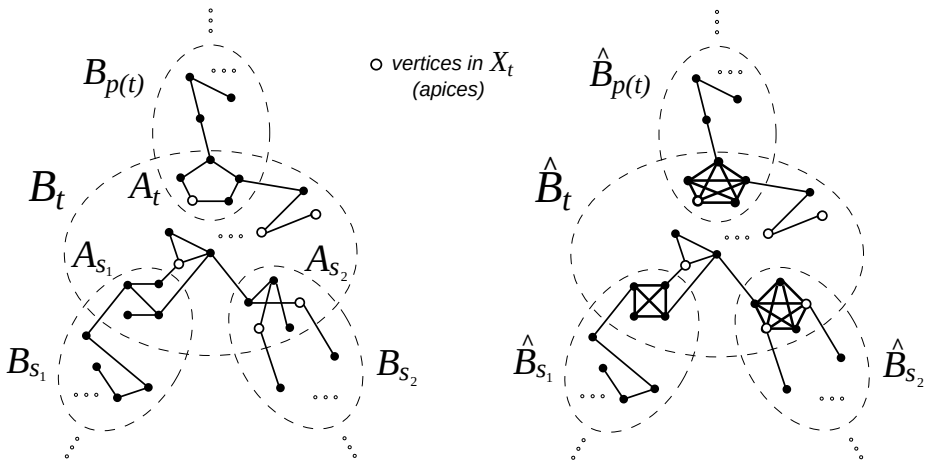

Figure 6.2: Tree-decomposition of a minor-free graph. The vertices in X_t (i.e., the *apices*) are depicted by $\circ$. Note that B_{s_1} and B_{s_2} could have non-empty intersection (in B_t).

Theorem 6.4 ([92, 187]) *For every graph M, there exists an integer h, depending only on the size of M, such that every graph excluding M as a minor can be obtained by clique sums of order at most h from graphs that can be h-nearly embedded in a surface Σ in which M cannot be embedded. Furthermore, such a clique decomposition can be found in polynomial time.*

Let G be an M-minor-free graph, and let $(T, \mathcal{B} = \{B_t\})$ be a clique decomposition of G given by Theorem 6.4. We suppose in addition that T is rooted at a given vertex $r \in V(G)$. We define $A_t := B_t \cap B_{p(t)}$ where $p(t)$ is the unique parent of the vertex t in T, and $A_r = \emptyset$. Let $\hat{B}_t$ be the graph obtained from B_t by adding all the possible edges between the vertices of A_t and also between the vertices of A_s, for each child s of t. In this way, A_t and A_s's will induce cliques in $\hat{B}_t$ (see Figure 6.2). In addition, G becomes an h-clique sum of the graphs $\hat{B}_t$ according to the above tree T where each $\hat{B}_t$ is h-nearly embeddable in a surface Σ in which M cannot be embedded. Let X_t be the set of apices of $\hat{B}_t$; we have $|X_t| \leq h$ and $\hat{B}_t \setminus X_t$ has linear local treewidth. We denote by G_t the subgraph induced by all the vertices of $B_t \cup \bigcup_s B_s$, for s ranging over all descendants of t in T.

In order to simplify the presentation, in what follows, we will restrict ourselves to the case $d = 3$, but it is quite straightforward to check that the proof extends to all $d \geq 3$. Recall that we are looking for a subset of vertices S, of size at most k, which induces a graph $H = G[S]$ of minimum degree at least three.

Our algorithm consists of two levels of dynamic programming. The top level of dynamic programming runs over the clique decomposition, and within each subproblem of this dynamic programming, we focus on the induced subgraph of the vertices in B_t. Our first level of dynamic programming computes the size of a smallest subgraph of G_t, complying with degree constraints on the vertices of A_t. These constraints, as before, represent the

degree of each vertex of A_t in the subgraph $H_t := G_t[S_t]$, i.e., the *trace* of H in G_t, where $S_t = S \cap V(G_t)$. This two-level dynamic programming requires a combinatorial bound on the treewidth as a function of the parameter k for each of the B_t's (after removing the apices X_t from B_t). The next two lemmas are used later to obtain this combinatorial bound.

Lemma 6.2 *Let $H = G[S]$ be a connected induced subgraph of G. Then the subgraph $\hat{B}_t[S \cap B_t]$ is connected.*

The proof of Lemma 6.2 easily follows from the properties of a tree-decomposition and the fact that A_t and A_s's are cliques in $\hat{B}_t$, for s a child of t in T.

Lemma 6.3 *Let $H = G[S]$ be a smallest connected subgraph of G of minimum degree at least three. Then the subgraph $\hat{B}_t[S_t \cap B_t \setminus X_t]$ has at most $3h + 1$ connected components, where h is the integer given by Theorem 6.4.*

Proof: Let $C_1, \ldots, C_r$ be the connected components of $L := \hat{B}_t[S_t \cap B_t \setminus X_t]$. We want to prove that $r \leq 3h + 1$. Assume for the sake of a contradiction that $r > 3h + 1$. We will find another solution H' with size strictly smaller than H, which will contradict our assumption that H is of minimum size.

The graph H' is defined as follows. For each vertex $v \in X_t \cap S_t$, let

$$b_v := \min\{d_{H_t}(v), 3\}.$$

Then for each vertex $v \in X_t \cap S_t$, we choose at most b_v connected components of L, covering at least b_v neighbors of v in H_t. We also add the connected component containing all the vertices of $A_t \setminus X_t$ (recall that A_t induces a clique in $\hat{B}_t$). Let A be the union of all the vertices of these connected components. Since $|X_t| \leq h$, A has at most $3h + 1$ connected components. Also, since A_s induces a clique in $\hat{B}_t$, for each child s of t such that $A_s \cap A \neq \emptyset$, we have that $A_s \setminus X_t \subset A$. We define H' as follows.

$$H' := G\left[\left(\bigcup_{\{s \,:\, A_s \cap A \neq \emptyset\}} S_s\right) \cup ((X_t \cup A) \cap S_t) \cup (S \setminus S_t)\right].$$

Clearly, $H' \subseteq H$. We have that $|H'| < |H|$ because, assuming that $r > 3h+1$, there are some vertices of $H_t \subset H$ which are in some connected component C_i which does not intersect H'.

Thus, it just remains to prove that H' is indeed a solution of kSMD3, i.e., H' has minimum degree at least 3. We prove it using a sequence of four simple claims:

Claim 6.4 *The degree of each vertex $v \in (V(H') \cap X_t)$ is at least 3 in H'.*

Proof: This is because each such vertex v has degree at least b_v in H'_t. If $d_v < 3$, then v should be in A_t (if not, v has degree $d_v < 3$ in H, which is impossible), hence v is connected to at least $3 - d_v$ vertices in $S \setminus S_t$. But $S \setminus S_t$ is included in H', and so every vertex of $X_t \cap V(H')$ has degree at least 3 in H'. □

Claim 6.5 *The degree of each vertex in* $(H \setminus H_t)$ *is at least* 3 *in* H'.

Proof: This follows because $A_t \cap H \subset H'$. □

Claim 6.6 *The degree of each vertex in* A *is at least* 3 *in* H'.

Proof: Every vertex in A has the same degree in both H' and H. This is because A is the union of some connected components, and no vertex of A is connected to any other vertex in any other component. □

Claim 6.7 *Every other vertex of* H' *also has degree at least* 3.

Proof: To prove the claim we prove that the vertices of $H' \setminus (G[X_t] \cup (H \setminus H_t) \cup A)$ have degree at least 3 in H'. Remember that all these vertices are in some S_s, for some s such that A_s has a non-empty intersection with A. We claim that all these vertices have the same degree in both H and H'. To prove this, note that $H' \cap A_s = H \cap A_s$ for all such s. Indeed, $(A_s \setminus X_t) \subset A$, and so $A_s \subset (A \cup X_t)$. Let u be such a vertex. We can assume that $u \notin X_t$. If $u \in A_s$, then clearly $u \in A$, and we are done. If $u \in (S_s \setminus A)$, then every neighbor of u is in H_s. But $H_s \subset H'$, hence we are also done in this case. □

This concludes the proof of the lemma. □

We define a *coloring* of A_t to be a function $c : A_t \cap S \to \{0, 1, 2, 3\}$. For $i < 3$, $c(v) = i$ means that the vertex v has degree i in the subgraph H_t of G_t that we are looking for, and $c(v) = 3$ means that v has degree at least three in H_t. By $a(t, c)$ we denote the minimum size of a subgraph of G_t with the prescribed degrees in A_t according to c. We describe in what follows the different steps of our algorithm.

Recursively, starting from the leaves of T and moving towards the root, for each node $t \in V(T)$ and for every coloring c of A_t, we compute $a(t, c)$ from the values of $a(s, c)$, where s is a child of t, or we store $a(t, c) = +\infty$ if no such subgraph exists. The steps involved in computing $a(t, c)$ for a fixed coloring c are the following:

(i) We guess a subset $R_t \subseteq X_t \setminus A_t$ such that $R_t \subseteq S_t$. We have at most 2^h choices for R_t.

(ii) For each vertex v in R_t, we guess whether v is adjacent to a vertex of $B_t \setminus (R_t \cup A_t)$, i.e., we test all the 2-colorings $\gamma : R_t \to \{0, 1\}$; a coloring has the following meaning: $\gamma(v) = 1$ if and only if v is adjacent to a vertex of $B_t \setminus (R_t \cup A_t)$. The number of such colorings is at most 2^h. Let γ be a fixed coloring. For each of the vertices v in R_t with $\gamma(v) = 1$, we guess one vertex in $B_t \setminus (R_t \cup A_t)$, which we suppose to be in S_t. For each coloring γ, we have at most n^h choices for the new vertices which could be included in S_t. If a vertex has $\gamma(v) = 0$, it is not allowed to be adjacent to any vertex of B_t besides the vertices in $A_t \cup R_t$. Let D_t^γ be the chosen vertices at this level.

(iii) We remove now all the vertices of X_t from B_t. Lemma 6.3 ensures that the induced graph $\hat{B}_t[S_t \cap B_t \setminus X_t]$ has at most $3h+1$ connected components. We then choose these connected components of $\hat{B}_t[S_t \cap B_t \setminus X_t]$ by guessing a vertex from these connected components in $B_t \setminus X_t$. Since we need to choose at most $3h+1$ vertices this way, we have at most $(3h+1)n^{3h+1}$ new choices. Let these newly chosen vertices be F_t^γ and

$$R_t^\gamma = R_t \cup D_t^\gamma \cup F_t^\gamma \cup \{v \in A_t \setminus X_t \mid c(v) \neq 0\}.$$

Let G_t^* be the graph induced by the k-neighborhood (vertices at distance at most k) of all vertices of R_t^γ in $\hat{B}_t \setminus X_t$, i.e., $G_t^* = (\hat{B}_t \setminus X_t)[N^k(R_t^\gamma)]$.

(iv) Each connected component of G_t^* has diameter at most $2k$ in $\hat{B}_t \setminus X_t$. As $\hat{B}_t \setminus X_t$ has bounded local treewidth, this implies that G_t^* has treewidth bounded by a function of k. By the result of Demaine and Hajiaghayi [93], this function can be chosen to be linear.

(v) In this step, we first find a tree-decomposition $(\mathcal{T}_\gamma, \{U_p\})$ of G_t^*. Since $A_s \cap G_t^*$ is a clique, it appears in a bag of this tree-decomposition. Let p be the node representing this bag in $\mathcal{T}_\gamma$. We create now a new bag containing the vertices of $A_s \cap G_t^*$, and modify $\mathcal{T}_\gamma$ by adding a leaf connected to p which contains this new bag. With slight abuse of notation, we call this new decomposition $\mathcal{T}_\gamma$ and denote by s this distinguished leaf containing the bag $A_s \cap G_t^*$. We also add all the vertices of A_t to all the bags of this tree-decomposition, increasing the bag size by at most h. Now we apply a dynamic programming algorithm similar to the one we used for the bounded local treewidth case. Remember that for each child s of t, we have a leaf in this (new) decomposition with the bag $A_s \cap G_s^*$. The aim is to find an induced subgraph of minimum size which respects all the choices we have made earlier.

We start from the leaves of $\mathcal{T}_\gamma$ and move towards its root. At this point we have all the values of $a(s, c')$ for all possible colorings c' of A_s, where s is a child of t (because of the first level of dynamic programming). To compute $a(t, c)$ we apply the dynamic programming algorithm of Lemma 6.1 with the restriction that for each *distinguished* leaf s of this decomposition, we already have all the values $a(s, c)$ for all colorings of $A_s \cap G_s^*$ (we extend this coloring to all A_s by giving the zero values to the vertices of $A_s \setminus G_s^*$). Note that the only difference between this dynamic programming and the one of Lemma 6.1 is the way we initialize the leaves of the tree.

(vi) Among all the subgraphs we found in this way, we keep the minimum size of a subgraph with the degree constraint c on A_t. Let $a(t, c)$ be this minimum.

(vii) If for some vertex t and a coloring $c : A_t \to \{0, 3\}$, we have $1 \leq a(t, c) \leq k$, the algorithm return YES, meaning that the graph contains a subgraph of size at most k and minimum degree at least three. If not, we conclude that such a subgraph does not exist.

This completes the description of the algorithm. Now we discuss the time complexity of this algorithm. Let C_M be the constant determining the linear local treewidth of the surfaces in which M cannot be embedded. For each fixed coloring c, we need time $4^{C_M k}(C_M k+1)^9 n^{4h+1}$ to obtain $a(t, c)$, where $t \in T$. Since the number of colorings of each A_t is at most 4^h, and the size of the clique decomposition is $O(n)$, we get the following theorem:

Theorem 6.5 *Let $\mathcal{C}$ be the class of graphs with excluded minor M. Then, for any graph in $\mathcal{C}$, one can find an induced subgraph of size at most k with degree at least 3 in time $O(4^{O(k+h)}(O(k))^9 n^{O(1)})$, where the constants in the exponents depend only on M.*

Theorem 6.5 can be generalized to larger values of d with slight modifications. We have the following theorem:

Theorem 6.6 *Let $\mathcal{C}$ be a class of graphs with an excluded minor M. Then, for any graph in $\mathcal{C}$, one can find an induced subgraph of size at most k with degree at least d in time $O((d+1)^{O(k+h)}(O(k))^{d^2} n^{O(1)})$, where the constants in the exponents depend only on M.*

6.4 Conclusions

In this chapter we studied the parameterized complexity of the following two problems: given two positive integers d and k, finding a d-regular (induced or not) subgraph with at most k vertices, and finding a subgraph with at most k vertices and of minimum degree at least d.

We first showed that both problems are fixed-parameter intractable in general graphs. More precisely, we proved that the first problem is $W[1]$-hard using a reduction from Multi-Color Clique. The hardness of the second problem followed from an extension of an already know result. We then provided explicit fixed-parameter tractable (FPT) algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach. These algorithms are considerably faster than those coming from the meta-theorem of Frick and Grohe [128] about problems definable in first-order logic over "locally tree-decomposable structures".

Finally, note that the parameterized tractability of the kSMDd problem for the case $d = 3$ remains open. We conjecture that:

Conjecture 6.1 kSMD3 *is* $W[1]$*-hard.*

Chapter 7

Subexponential Parameterized Algorithms on Planar Graphs

In this chapter we present subexponential parameterized algorithms on planar graphs for a family of problems that consist in, given a graph G, finding a connected (induced) subgraph H with bounded maximum degree, while maximizing the number of edges (or vertices) of H. These problems are natural generalizations of Longest Path. Our approach uses bidimensionality theory combined with novel dynamic programming techniques over branch decompositions of the input graph. These techniques can be applied to a more general family of problems that deal with finding connected subgraphs under certain degree constraints.

Keywords: parameterized complexity, planar graphs, subexponential algorithm, branch decomposition, graph minors, bidimensionality, Catalan structures.

7.1 Introduction

During the last years a considerable amount of work has been devoted to design subexponential parameterized algorithms for NP-hard optimization problems on planar graphs and, more generally, on sparse classes of graphs [95, 98–102, 144]. In this chapter we apply the general approach of [95, 98–102, 144] to a family of problems dealing with finding connected subgraphs under degree constraints. Along the way, we introduce novel dynamic programming techniques over branch decompositions that can be applied to more general classes of problems.

We define the following family of problems for $d \geq 2$.

MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$)
Input: A graph G and a non-negative integer k.
Question: Does G contain a connected subgraph H with maximum degree at most d and at least k edges?

If $d = 2$ the problem is equivalent to the LONGEST PATH (or CYCLE, if G is Hamiltonian) problem, hence MDBCS$_d$ is a generalisation of it. MDBCS$_d$ is one of the classical NP-hard problems listed in [134], and we have seen in Chapter 5 that it is not in APX for any $d \geq 2$. Without the connectivity constraint, the problem is known to be in P using matching techniques [165]. When the problem is parameterized by k we denote it by k-MDBCS$_d$. (We refer to [123] for an introduction to parameterized complexity.) Our target is to find $2^{O(\sqrt{k})} \cdot O(n)$ step algorithms to solve it when the input is restricted to planar graphs. Section 7.3 is devoted to obtain combinatorial bounds using bidimensionality theory. Section 7.4 presents new dynamic programming techniques, that can be applied to general graphs. In Section 7.5 we see how to speed-up these algorithms when the input is restricted to planar graphs, using Catalan structures. This strategy can be extended to several related problems asking for a maximum connected subgraph satisfying certain degree constraints, as discussed in Section 7.6. Some open problems are listed in Section 7.7. We first provide some background in Section 7.2.

7.2 Background

Recall the definition of branchwidth from Section I.1.1 (page 22) and the definition of graph minor from Section I.1.2 (page 22). Recall also that a parameter **P** defined on simple undirected graphs is *closed under taking of minors* (or simply *minor closed*) if $G' \preceq G \Rightarrow \mathbf{P}(G') \leq \mathbf{P}(G)$ (here "$\preceq$" denotes the minor relation). The following fundamental theorem states that square grids are the obstruction for branchwidth on planar graphs.

Theorem 7.1 (Robertson, Seymour, and Thomas [185]) *Let $\ell \geq 1$ be an integer. Every planar graph of branchwidth at least ℓ contains an $(\lfloor \ell/4 \rfloor \times \lfloor \ell/4 \rfloor)$-grid as a minor.*

A parameter **P** is *minor bidimensional* [94] with *density* δ if

- P is minor closed, and
- for the $(r \times r)$-grid R, $P(R) = (\delta r)^2 + o((\delta r)^2)$.

Theorem 7.1 implies the following useful property.

Lemma 7.1 (Demaine et al. [94]) *If* **P** *is a bidimensional parameter with density δ then for any planar graph G, $\mathbf{bw}(G) \leq \frac{4}{\delta} \cdot \sqrt{\mathbf{P}(G)} + O(1)$.*

There is a recent and powerful theory about bidimensional parameters, called *bidimensionality theory*, that has proved very useful in the design of subexponential exact and parameterized algorithms for many hard problems [95, 98–102, 144].

7.3 Bounds for Branchwidth

We define the following parameter on simple undirected graphs.

$$\textbf{medbcs}_d(G) = \max\{|E(H)| \mid H \subseteq G \ \wedge \ H \text{ is connected } \wedge \ \Delta(H) \leq d\}.$$

Lemma 7.2 *For any integer $d \geq 1$, the parameter* $\textbf{medbcs}_d$ *is minor closed.*

Proof: If G' occurs from G after an edge removal, then clearly $\textbf{medbcs}_d(G') \leq \textbf{medbcs}_d(G)$. Let us see that the same holds if G' occurs from G after the contraction of an edge $\{x,y\}$. Indeed, we shall see that given any connected subgraph $H' \subseteq G'$ with $\Delta(H') \leq d$, we can find a connected subgraph $H^* \subseteq G$ with $\Delta(H^*) \leq d$ and $|E(H^*)| \geq |E(H')|$. Let H be the major of H' in G. We can assume that $v_{xy} \in V(H')$, otherwise we set $H^* = H$. We define $N_{xy} = N_H(x) \cap N_H(y)$, $N_{x-y} = N_H(x) - N_{xy} - \{y\}$, and $N_{y-x} = N_H(y) - N_{xy} - \{x\}$. The subgraph H is connected and $|E(H)| \geq |E(H')|$, but the vertices x, y, and those in N_{xy} may have degree $d+1$. Since $\Delta(H') \leq d$, also $|N_{H'}(v_{xy})| = |N_{x-y}| + |N_{y-x}| + |N_{xy}| \leq d$. Suppose w.l.o.g. that $|N_{x-y}| \geq |N_{y-x}|$. We distinguish several cases to define the subgraph H^*: If $|N_{x-y}| = d$, let $H^* = (V(H) - \{y\}, E(H) - \{x,y\})$. Suppose henceforth that $|N_{x-y}| < d$. If $|N_{xy}| = 0$, let $H^* = H$. If $N_{xy} = \{z_1\}$, let $H^* = (V(H), E(H) - \{x,z_1\})$. Finally, if $N_{xy} = \{z_1, \ldots, z_k\}$ for some $k \geq 2$, let $H^* = (V(H), E(H) - \{x,z_1\} - \cup_{i=2}^{k}\{y,z_i\})$. It is easy to check that, in all cases, the subgraph H^* is connected, $\Delta(H^*) \leq d$, and $|E(H^*)| \geq |E(H')|$. □

Using Lemmas 7.2 and 7.1 we can obtain a combinatorial bound of the parameter $\textbf{medbcs}_d$ in terms of the branchwidth of the planar graph G.

Lemma 7.3 *For any $d \geq 2$ and for any planar graph G it holds that*

$$\textbf{bw}(G) \leq \frac{4}{\delta} \cdot \sqrt{\textbf{medbcs}_d(G)} + O(1), \quad \textit{with} \quad \delta = \begin{cases} 1 & \text{, if } d = 2 \\ \sqrt{3/2} & \text{, if } d = 3 \\ \sqrt{2} & \text{, if } d \geq 4 \end{cases}$$

Proof: We shall prove that the parameter $\textbf{medbcs}_d(G)$ is bidimensional for any $d \geq 2$. It is minor closed due to Lemma 7.2. Let us see how the parameter behaves on the grid. Let R be an $(r \times r)$-grid. If $d = 2$, then clearly $\textbf{medbcs}_2(R) \geq r^2 - 1$ (or r^2 if r is even, because in this case the grid contains a Hamiltonian cycle). That is, the density of $\textbf{medbcs}_2$ is 1. If $d \geq 4$ then the optimal solution contains all the edges, i.e., $\textbf{medbcs}_d(R) = 2r(r-1)$. Said otherwise, the density is $\sqrt{2}$. Finally, if $d = 3$, we shall see that $\textbf{medbcs}_3(R) \geq 2r(r-1) - \left\lceil \frac{r-2}{2} \right\rceil (r-2)$. Such a solution is obtained in the following way. Take all the *horizontal* edges of the grid, and the *vertical* edges corresponding to the first and the last column. Then, beginning from the first row, take alternatively the remaining vertical edges (see Figure 7.1 for an illustration). One can easily check that the subgraph obtained in this way is connected, has maximum degree 3 and has $2r(r-1) - \left\lceil \frac{r-2}{2} \right\rceil (r-2)$ edges. That is, the density of $\textbf{medbcs}_3$ is at least $\sqrt{3/2}$. The result follows from Lemma 7.1. □

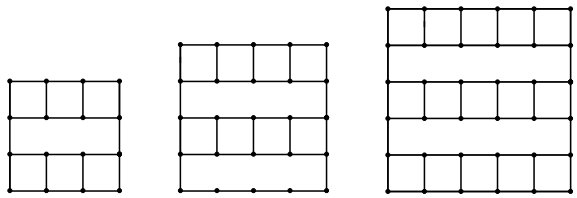

Figure 7.1: Connected subgraphs with maximum degree 3 on (4×4), (5×5), and (6×6)-grids respectively, used in the proof of Lemma 7.3.

7.4 The Algorithms

Let G be in this section a (not necessarily planar) graph on n vertices. We denote the *empty set* by $\emptyset$ and the *empty function* by $\varnothing$. Let (T, μ) be a branch decomposition of width $\leq \ell$ of G. We pick an arbitrary edge $e^* \in E(T)$, we subdivide it by adding a new vertex v_{new} and then add a new vertex r and make it adjacent to v_{new}. We extend μ by setting $\mu(r) = \emptyset$ and we root T at vertex r. For each $e \in E(T)$ let T_e be the tree of the forest $T \backslash e$ that does not contain r as a leaf (i.e., the tree that is "below" e in the rooted tree T) and let E_e be the edges that are images, via μ, of the leaves of T that are also leaves of T_e. We denote $G_e = G[E_e]$. Observe that, if $e_r = \{v_{\text{new}}, r\}$, then $G_{e_r} = G$.

Given a set A, we define a *d-weighted packing* of A as any pair $(\mathcal{A}, \psi)$ where $\mathcal{A}$ is a (possible empty) collection of mutually disjoint nonempty subsets of A and $\psi : A \to \{0, \ldots, d\}$ is a mapping corresponding integers from 0 to d to the elements of A. It will be convenient to think of such a packing $\mathcal{A}$ of A as a hypergraph $\mathcal{G} = (A, \mathcal{A})$. Note that, by definition, $\mathcal{A}$ is a matching in $\mathcal{G}$. For convenience, given such a collection $\mathcal{A}$, we denote by $\cup\mathcal{A}$ the set $\bigcup_{X \in \mathcal{A}} X$.

Let $(\mathcal{A}, \psi)$ and $(\mathcal{A}', \psi')$ be two d-weighted packings of two sets A and A'. We define $(\mathcal{A}, \psi) \oplus (\mathcal{A}', \psi')$ as the $2d$-weighted packing $(\mathcal{A}'', \psi'')$ of $A'' = A \cup A'$ where $\mathcal{A}''$ is the packing of A'' defined by the connected components of the hypergraph $(A \cup A', \mathcal{A} \cup \mathcal{A}')$ (i.e., the nonempty subsets of the packing $\mathcal{A}''$ are the vertex sets corresponding to the connected components of the hypergraph $(A \cup A', \mathcal{A} \cup \mathcal{A}')$) and where for any $x \in A \cup A'$,

$$\psi''(x) = \begin{cases} \psi(x) & \text{, if } x \in A - A' \\ \psi(x) + \psi'(x) & \text{, if } x \in A \cap A' \\ \psi'(x) & \text{, if } x \in A' - A \end{cases}$$

If $(\mathcal{A}, \psi)$ is a d-weighted packing of a set A and $A' \subseteq A$, we define $(\mathcal{A}, \psi)|_{A'}$ as the d-weighted packing $(\mathcal{A}', \psi')$ of the set A' where $\mathcal{A}' = \{X \cap A' \mid X \in \mathcal{A}\}$ and $\psi' = \{(x, \psi(x)) \mid x \in A'\}$.

Let $\mathscr{P}_e$ be the collection of all d-weighted packings $(\mathcal{A}, \psi)$ of $\mathbf{mid}(e)$, and let $\ell = |\mathbf{mid}(e)|$. Observe that if $e_r = \{v_{\text{new}}, r\}$, then $\mathscr{P}_{e_r} = \{(\emptyset, \varnothing)\}$. We use the notation $C(H)$ for the set of

connected components of a graph (or hypergraph) H. Given $(\mathcal{A}, \psi) \in \mathscr{P}_e$ we define

$$\begin{aligned}\mathbf{opt}_e(\mathcal{A},\psi) \;=\; \max\{\{0\} \cup \{|E(H)| \;:\; & \exists\, H \subseteq G_e : \Delta(H) \leq d \;\wedge \\ & \text{if } (\mathcal{A} \neq \emptyset) \text{ then} \\ & \quad \{V(H') \cap \mathbf{mid}(e) \mid H' \in \mathcal{C}(H)\} = \mathcal{A} \;\wedge \\ & \quad \{(v, \mathbf{deg}_H(v)) \mid v \in \cup_{A\in\mathcal{A}} A\} = \psi \\ & \text{else if } (\mathcal{A} = \emptyset) \text{ then} \\ & \quad |\mathcal{C}(H)| \leq 1 \;\wedge\; V(H) \cap \mathbf{mid}(e) = \emptyset \;\}\}\end{aligned}$$

Clearly, $\mathbf{opt}_{e_r}(\emptyset, \varnothing) = \mathbf{medbcs}_d(G)$. The idea is the following:

- If $\mathcal{A} \neq \emptyset$, we look for the best solution H in the graph G_e such that its restriction to $\mathbf{mid}(e)$ induces the connected components given by $\mathcal{A}$ and obeys the degrees given by ψ.

- Otherwise, if $\mathcal{A} = \emptyset$, we look for the best solution H in G_e not intersecting $\mathbf{mid}(e)$. Since $\mathbf{mid}(e)$ is a separator of G, it is clear that in this case the solution H must be a connected subgraph of G_e disjoint from $\mathbf{mid}(e)$.

Let us now see how these values of $\mathbf{opt}_e(\mathcal{A}, \psi)$ can be explicitly computed using dynamic programming over a branch decomposition of G.

Let e, e_1, e_2 be three edges of T that are incident to the same vertex and such that e is closer to the root of T than the other two (see the upper part of Figure 7.2). To perform the *join/forget* operations in the middle set $\mathbf{mid}(e)$, we distinguish two cases according to the packing $\mathcal{A}$ of $\mathbf{mid}(e)$:

(1) In the case $\mathcal{A} \neq \emptyset$, the value of $\mathbf{opt}_e(\mathcal{A}, \psi)$ is given by

$$\begin{aligned}\mathbf{opt}_e(\mathcal{A},\psi) \;=\; \max\{\{0\} \cup \{l : \exists (\mathcal{A}_i, \psi_i) \in \mathscr{P}_{e_i}, i = 1,2, \text{ such that} \\ \cup\, \mathcal{A}_1 \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) = \cup \mathcal{A}_2 \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) \wedge \\ (\mathcal{A}_1, \psi_1) \oplus (\mathcal{A}_2, \psi_2) \text{ is a } d\text{-weighted} \\ \quad \text{packing of } \mathbf{mid}(e_1) \cup \mathbf{mid}(e_2) \;\wedge \\ (\mathcal{A}, \psi) = ((\mathcal{A}_1, \psi_1) \oplus (\mathcal{A}_2, \psi_2))|_{\mathbf{mid}(e)} \;\wedge \\ \text{if } (\mathcal{A}_1 = \emptyset) \text{ then } l = \mathbf{opt}_{e_2}(\mathcal{A}_2, \psi_2) \\ \text{if } (\mathcal{A}_2 = \emptyset) \text{ then } l = \mathbf{opt}_{e_1}(\mathcal{A}_1, \psi_1) \\ \text{else } l = \mathbf{opt}_{e_1}(\mathcal{A}_1, \psi_1) + \mathbf{opt}_{e_2}(\mathcal{A}_2, \psi_2) \;\}\}\end{aligned}$$

(2) In the case $\mathcal{A} = \emptyset$, the value of $\mathbf{opt}_e(\emptyset, \psi)$ is given by

$$\begin{aligned}
\mathbf{opt}_e(\emptyset,\psi) \;=\; \max\{\{0\} \cup \{l : \exists (\mathcal{A}_i,\psi_i) \in \mathscr{P}_{e_i}, i=1,2, \text{ such that} \\
\cup\mathcal{A}_1 \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) = \cup\mathcal{A}_2 \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) \wedge \\
(\mathcal{A}_1,\psi_1) \oplus (\mathcal{A}_2,\psi_2) \text{ is a } d\text{-weighted} \\
\quad \text{packing of } \mathbf{mid}(e_1) \cup \mathbf{mid}(e_2) \;\wedge \\
(\emptyset,\psi) = ((\mathcal{A}_1,\psi_1) \oplus (\mathcal{A}_2,\psi_2))|_{\mathbf{mid}(e)} \;\wedge \\
\text{if } (\mathcal{A}_1 = \emptyset \;\wedge\; \mathcal{A}_2 = \emptyset) \text{ then} \\
\quad l = \max\{\mathbf{opt}_{e_1}(\mathcal{A}_1,\psi_1), \mathbf{opt}_{e_2}(\mathcal{A}_2,\psi_2)\} \\
\text{if } (\mathcal{A}_1 \neq \emptyset \;\wedge\; \mathcal{A}_2 = \emptyset) \text{ then} \\
\quad l = \max\{\mathbf{opt}_{e_2}(\mathcal{A}_2,\psi_2), \{\mathbf{opt}_{e_1}(\mathcal{A}_1,\psi_1)|_X : X \in \mathcal{A}_1\}\} \\
\text{if } (\mathcal{A}_1 = \emptyset \;\wedge\; \mathcal{A}_2 \neq \emptyset) \text{ then} \\
\quad l = \max\{\mathbf{opt}_{e_1}(\mathcal{A}_1,\psi_1), \{\mathbf{opt}_{e_2}(\mathcal{A}_2,\psi_2)|_X : X \in \mathcal{A}_2\}\} \\
\text{if } (\mathcal{A}_1 \neq \emptyset \;\wedge\; \mathcal{A}_2 \neq \emptyset) \text{ then} \\
\quad l = \max\{\mathbf{opt}_{e_1}(X,\psi_1)|_{\mathbf{mid}(e_1)} + \mathbf{opt}_{e_2}(X,\psi_2)|_{\mathbf{mid}(e_2)} : \\
\qquad X \in C(\mathbf{mid}(e_1) \cup \mathbf{mid}(e_2), \mathcal{A}_1 \cup \mathcal{A}_2)\}\;\}\}
\end{aligned}$$

These ideas are schematically illustrated in Figure 7.2. Finally, suppose that $e_{\text{leaf}} = \{x, y\} \in E(T)$ is an edge such that either x or y is a leaf of T. Let $\{v_1, v_2\} \in E(G)$ be the image under μ of the endpoint of e which is a leaf of T. Then

$$\mathbf{opt}_{e_{\text{leaf}}}(\mathcal{A},\psi) = \begin{cases} 1 & \text{, if } (\mathcal{A} = \{\{v_1, v_2\}\} \quad \wedge \quad \psi = \{(v_1,1),(v_2,1)\}) \\ 0 & \text{, otherwise} \end{cases}$$

Running time. The size of the tables of the dynamic programming over the branch decomposition of the input graph, namely $|\mathscr{P}_e|$, determines the running time of our algorithms. The number of ways a set of ℓ elements can be partitioned into nonempty subsets is well-known as the ℓ-th *Bell number* [117] and is denoted by B_ℓ. We can express $|\mathscr{P}_e|$ in terms of the Bell numbers:

$$|\mathscr{P}_e| \;=\; (d+1)^\ell \cdot \sum_{i=0}^{\ell} \binom{\ell}{i} B_{\ell-i} \;\leq\; (d+1)^\ell \cdot 2^{2\ell \cdot \log \ell}, \tag{7.1}$$

where the last inequality is an easy exercise using that $B_\ell \leq \frac{e^\ell - 1}{(\log \ell)^\ell} \ell!$ [117]. At each edge e of the branch decomposition, to compute all the values $\mathbf{opt}_e(\mathcal{A},\psi)$ we test all the possibilities of combining d-weighted packings of the two middle sets $\mathbf{mid}(e_1)$ and $\mathbf{mid}(e_2)$. The operations $(\mathcal{A}_1,\psi_1) \oplus (\mathcal{A}_2,\psi_2)$ and $(\mathcal{A},\psi)|_{A'}$ take $O(|\mathbf{mid}(e)|)$ time. Let $m = |E(G)|$. Hence, by Equation (7.1), given a branch decomposition of a general graph G of width at most ℓ, the value of $\mathbf{medbcs}_d(G)$ can be computed in $(d+1)^{2\ell} \cdot 2^{4\ell \cdot \log \ell} \cdot \ell \cdot m$ steps.

7.5 Speed-up for Planar Graphs using Catalan Structures

In this section we will see that when the input is restricted to planar graphs the term $2^{O(\ell \cdot \log \ell)}$ in Equation (7.1) can be reduced to $2^{O(\ell)}$. We need first some definitions.

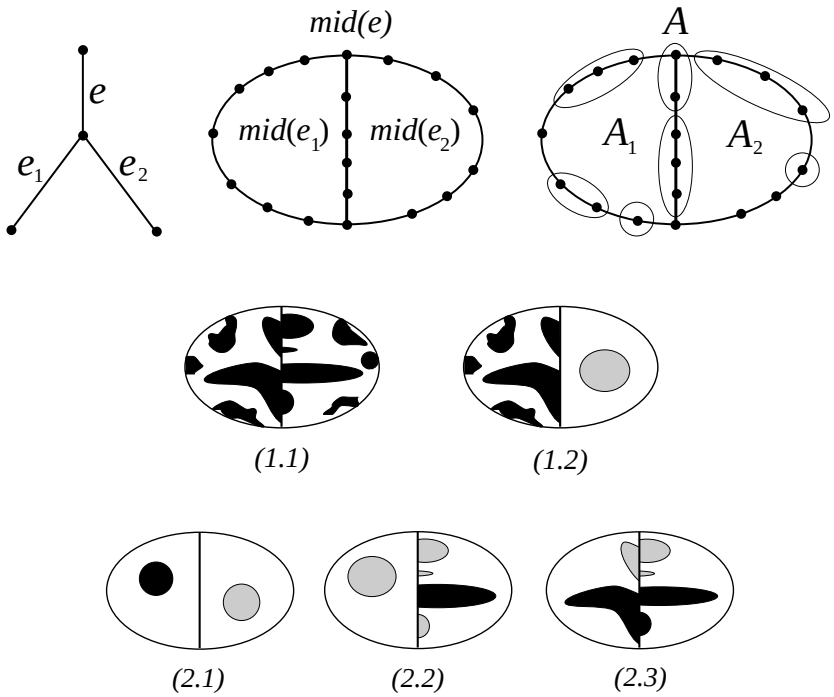

Figure 7.2: *Join/forget* operations in the dynamic programming over a branch decomposition. The dark regions represent an optimal subgraph in each case. Case *(1)*: $\mathcal{A} \neq \emptyset$; *(1.1)* $\mathcal{A}_1 \neq \emptyset, \mathcal{A}_2 \neq \emptyset$; *(1.2)* $\mathcal{A}_1 \neq \emptyset, \mathcal{A}_2 = \emptyset$. Case *(2)*: $\mathcal{A} = \emptyset$; *(2.1)* $\mathcal{A}_1 = \emptyset, \mathcal{A}_2 = \emptyset$; *(2.2)* $\mathcal{A}_1 = \emptyset, \mathcal{A}_2 \neq \emptyset$; *(2.3)* $\mathcal{A}_1 \neq \emptyset, \mathcal{A}_2 \neq \emptyset$.

Let G be a planar graph embedded on a sphere $\mathbb{S}$. An *O-arc* is a subset of $\mathbb{S}$ homeomorphic to a circle. An O-arc in S is called a *noose* of the embedding of G if it meets G only in vertices. A *sphere cut decomposition* or *sc-decomposition* (T, μ, π) of G is a branch decomposition of G with the following property: for every edge e of T, there exists a noose O_e meeting every face at most once and bounding the two open discs Δ_1 and Δ_2 such that $G_i \subseteq \Delta_i \cup O_e$, $1 \leq i \leq 2$. Thus O_e meets G only in $\mathbf{mid}(e)$ and its length is $|\mathbf{mid}(e)|$. A *clockwise traversal* of O_e in the embedding of G defines the cyclic ordering π of $\mathbf{mid}(e)$. We always assume that the vertices of every middle set $\mathbf{mid}(e) = V(G_1) \cap V(G_2)$ are enumerated according to π.

Theorem 7.2 (Seymour and Thomas [190]) *Let G be a planar graph of branchwidth at most ℓ without vertices of degree one embedded on a sphere. Then there exists an sc-decomposition of G of width at most ℓ.*

In addition, such an sc-decomposition can be constructed in time $O(n^3)$ [143].

The size of the tables of the dynamic programming algorithm is given by in how many ways a solution of k-MDBCS$_d$ in G_e can intersect $\mathbf{mid}(e)$. Let (T, μ, π) be a sphere cut decomposition of width $\leq \ell$, and we can assume $\ell \leq \mathbf{bw}(G)$ by Theorem 7.2. Then the

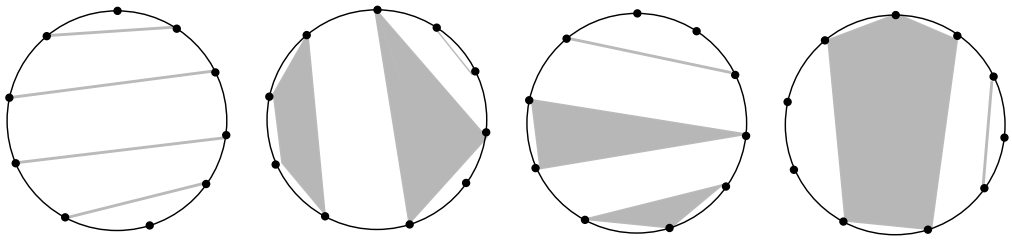

Figure 7.3: Catalan structures in the middle set of a sphere cut decomposition.

vertices of $\mathbf{mid}(e)$ are situated around a noose. A *non-crossing partition (ncp)* is a partition $P(n) = \{P_1, \ldots, P_m\}$ of the set $S = \{1, \ldots, n\}$ such that there are no numbers $a < b < c < d$ where $a, c \in P_i$, and $b, d \in P_j$ with $i \neq j$.

When we restrict the input graph G to be planar, then the subgraph given by the intersection of a partial solution of k-MDBCS$_d$ in G_e with $\mathbf{mid}(e)$ is also planar. The reduction from $2^{O(\ell \cdot \log \ell)}$ to $2^{O(\ell)}$ is based on calculating in how many ways we can draw hyperedges inside a cycle such that they touch the cycle on its vertices and they do not share common internal points in the plain (they do not intersect), as it is illustrated in Figure 7.3.

The number of such configurations is closely related to the number of *non-crossing partitions* over ℓ vertices, which is equal to the ℓ-th *Catalan number* $\mathrm{CN}(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \leq 4^\ell$ [162].

Indeed, in the same spirit of Equation (7.1) we can write

$$\begin{aligned} |\mathscr{P}_e| &= (d+1)^\ell \cdot \sum_{i=0}^{\ell} \binom{\ell}{i} \mathrm{CN}(\ell - i) \leq (d+1)^\ell \cdot \sum_{i=0}^{\ell} \binom{\ell}{i} 4^{\ell-i} = \\ &= (d+1)^\ell 4^\ell \cdot \sum_{i=0}^{\ell} \binom{\ell}{i} \left(\frac{1}{4}\right)^i = (d+1)^\ell 4^\ell \cdot \left(1 + \frac{1}{4}\right)^\ell = (d+1)^\ell \cdot 5^\ell. \end{aligned}$$

Since G is planar, $|E(G)| = O(|V(G)|)$, hence so is the number of middle sets in any branch decomposition of G. Therefore,

Proposition 7.1 *For every planar graph G and given a sphere cut decomposition (T, μ, π) of G of width $\leq \ell$, the value of $\mathbf{medbcs}_d(G)$ can be computed in $O\left((d+1)^{2\ell} \cdot 5^{2\ell} \cdot \ell \cdot n\right)$ steps.*

Let δ be the constant defined in Lemma 7.3. Summarizing,

Theorem 7.3 k-Planar Maximum d-Degree-Bounded Connected Subgraph *is solvable in time $O\left(2^{\log(5(d+1))8\sqrt{k}/\delta}\sqrt{k} \cdot n + n^3\right)$ for any $d \geq 2$.*

Proof: First, using Theorem 7.2, we construct in time $O(n^3)$ an optimal sphere cut decomposition of G of width $\mathbf{bw}(G)$. We distinguish two cases: If $\mathbf{bw}(G) > 4/\delta \cdot \sqrt{k}$, then by Lemma 7.3 the answer to the parameterized problem is automatically YES. Otherwise, $\mathbf{bw}(G) \leq 4/\delta \cdot \sqrt{k}$ and the value of $\mathbf{medbcs}_d(G)$ can be computed by Proposition 7.1 in time $O\left((d+1)^{8\sqrt{k}/\delta} \cdot 5^{8\sqrt{k}/\delta} \cdot 4/\delta\sqrt{k} \cdot n\right) = O\left(2^{\log(5(d+1))8\sqrt{k}/\delta}\sqrt{k} \cdot n\right)$. □

7.6 Extensions

Appropriate modifications of the dynamic programming algorithm of Section 7.4 allow us to obtain also subexponential parameterized algorithms for the variant of the problem in which the aim is to maximize the number of vertices of the subgraph H, as well as for the variant in which the output subgraph is required to be induced (for both the edge and vertex maximization versions). Another variant is when the list of prescribed degrees of the vertices belongs to a subset of $\mathbb{Z}_q$ for a fixed integer q. Finally, we discuss how to transform these parameterized algorithms into subexponential *exact* algorithms on planar graphs.

7.6.1 Maximizing the number of vertices

In this section we focus on the following family of problems for $d \geq 2$:

Vertex Maximum d-Degree-Bounded Connected Subgraph (VMDBCS$_d$)
Input: A graph G and a non-negative integer k.
Question: Does G contain a connected subgraph H with $\Delta(H) \leq d$ and $\lvert V(H)\rvert \geq k$?

In order to obtain subexponential parameterized algorithms for VMDBCS$_d$ on planar graphs, let us see how the techniques presented in the preceding sections must be modified. The corresponding parameter is

$$\mathbf{mvdbcs}_d(G) = \max\{|V(H)| \mid H \subseteq G \wedge H \text{ is connected} \wedge \Delta(H) \leq d\}.$$

First, it is easy to check that Lemmas 7.2 and 7.3 hold for the parameter $\mathbf{mvdbcs}_d(G)$ with $\delta = 1$ for any $d \geq 2$. Secondly, the dynamic programming approach of Section 7.4 remains the same, except for the following modifications.

When computing a partial solution $\mathbf{opt}_e(\mathcal{A}, \psi)$ in G_e from the partial solutions in G_{e_1} and G_{e_2}, we have to be careful in order to avoid counting twice the vertices that belong to both $\mathbf{mid}(e_1)$ and $\mathbf{mid}(e_2)$. More precisely,

- in the case $\mathcal{A} \neq \emptyset$, $\mathcal{A}_1 \neq \emptyset$, and $\mathcal{A}_2 \neq \emptyset$ we have that

$$l = \mathbf{opt}_{e_1}(\mathcal{A}_1, \psi_1) + \mathbf{opt}_{e_2}(\mathcal{A}_2, \psi_2) - |V(\mathcal{G}[\mathcal{A}_1]) \cap V(\mathcal{G}[\mathcal{A}_2])|,$$

  where $\mathcal{G}[\mathcal{A}_i]$, $i = 1, 2$, denotes the hypergraph induced by the hyperedges in $\mathcal{A}_i$; and

- in the case $\mathcal{A} = \emptyset$, $\mathcal{A}_1 \neq \emptyset$, and $\mathcal{A}_2 \neq \emptyset$ we have that

$$\begin{aligned} l \;=\; \max\{&\mathbf{opt}_{e_1}(X, \psi_1)|_{\mathbf{mid}(e_1)} + \mathbf{opt}_{e_2}(X, \psi_2)|_{\mathbf{mid}(e_2)} - \\ &|V(\mathcal{G}[X]) \cap \mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)| \;: \\ &X \in \mathcal{C}(\mathbf{mid}(e_1) \cup \mathbf{mid}(e_2), \mathcal{A}_1 \cup \mathcal{A}_2)\}\;\}. \end{aligned}$$

Also, if $e_{\text{leaf}} = \{x, y\} \in E(T)$ is an edge such that x is a leaf of T, and $\{v_1, v_2\} \in E(G)$ is the image of x under μ, then

$$\mathbf{opt}_{e_{\text{leaf}}}(\mathcal{A}, \psi) = \begin{cases} 2 & \text{, if } (\mathcal{A} = \{\{v_1, v_2\}\} \quad \wedge \quad \psi = \{(v_1, 1), (v_2, 1)\}) \\ & \quad \vee (\mathcal{A} = \{\{v_1\}, \{v_2\}\} \quad \wedge \quad \psi = \{(v_1, 0), (v_2, 0)\}) \\ 1 & \text{, if } (\mathcal{A} = \{\{v_1\}\} \quad \wedge \quad \psi = \{(v_1, 0), (v_2, 0)\}) \\ & \quad \vee (\mathcal{A} = \{\{v_2\}\} \quad \wedge \quad \psi = \{(v_1, 0), (v_2, 0)\}) \\ 0 & \text{, otherwise} \end{cases}$$

Finally, the speed-up described in Section 7.5 can be applied directly to VMDBCS$_d$, since Catalan structures also appear in the middle sets of a sc-decomposition of the planar input graph. Summarizing,

Theorem 7.4 *k-Planar Vertex Maximum d-Degree-Bounded Connected Subgraph is solvable in time $O\left(2^{\log(5(d+1))8\sqrt{k}}\sqrt{k} \cdot n + n^3\right)$ for any $d \geq 2$.*

7.6.2 Looking for an induced subgraph

It is also natural to ask, instead of for a subgraph H of the input graph G, for an *induced* subgraph H. In this section we focus on the edge-maximization version of the problem, the modifications for the node-maximization version being analogous to those described in Section 7.6.1. We denote the problem by Maximum d-Degree-Bounded Connected Induced Subgraph (MDBCIS$_d$).

In contrast to the dynamic programming presented in Section 7.4, now we need only to consider those packings $\mathcal{A}$ of $\mathbf{mid}(e)$ that "respect" the fact that the solution subgraph must be induced. For this, the only difference from the algorithms for the non-induced case is that the optimal solution in the leaves of the branch decomposition becomes

$$\mathbf{opt}_{e_{\text{leaf}}}(\mathcal{A}, \psi) = \begin{cases} 1 & \text{, if } (\mathcal{A} = \{\{v_1, v_2\}\} \quad \wedge \quad \psi = \{(v_1, 1), (v_2, 1)\} \\ 0 & \text{, otherwise} \end{cases}$$

Since for any middle set $\mathbf{mid}(e)$, $\mathscr{P}_e^{\text{ind}} \subseteq \mathscr{P}_e$, the running time of Theorem 7.3 also applies to k-Planar Maximum d-Degree-Bounded Connected Induced Subgraph, replacing the constant δ with $\delta/\sqrt{2}$ when $d \in \{2, 3\}$, due to the fact that the optimal subgraphs of MDBCIS$_d$ on the square grid (see Lemma 7.2) must be induced.

7.6.3 More general constraints on the degree

All the variants of the problem considered so far have in common that the degree of any vertex belonging to the output subgraph must lie in the interval $[0, d]$. It makes sense to consider a more general version in which the interval of allowed degrees depends on each vertex. Namely, for each vertex $v \in V(G)$ we are given an interval $I_v = [\ell_v, r_v]$ and we look for a maximum connected subgraph H in which the degree of each vertex v lies in I_v. (If $0 \in I_v$ then vertex v may not belong to $V(H)$.) When the output subgraph is not required to be connected, some variants of the problem are in P and some others

become NP-hard [165]. In general, we cannot guarantee that the parameters associated with this general problem are minor closed, hence the approach used with MDBCS$_d$ does not carry over. Nevertheless, we can obtain an algorithm to solve it similar to the one of Proposition 7.1, replacing the term $(d+1)^{2\ell}$ with $(\max_{v\in V(G)} r_v + 1)^{2\ell}$. The ideas behind the dynamic programming are essentially the same.

Another variant is obtained when forcing the allowed degrees to belong to a subset of $\mathbb{Z}_q$ for some fixed integer q. In this case it is not difficult to see that the term $(d+1)^{2\ell}$ can be replaced with $q^{2\ell}$. For instance, the case where all the degrees are required to be 0 (mod 2) corresponds to the Maximum Eulerian Subgraph problem. This approach, given a planar graph with a sphere cut decomposition of width $\leq \ell$, yields an algorithm to solve Maximum Eulerian Subgraph in time $O\left(2^{2\ell} \cdot 5^{2\ell} \cdot \ell \cdot n\right)$.

7.6.4 Exact algorithms

The subexponential parameterized algorithms we have presented on planar graphs can be naturally transformed to subexponential *exact* algorithms by using that for any planar graph G, $\mathbf{bw}(G) \leq \sqrt{4.5 \cdot |V(G)|}$ [125].

Indeed, given a planar graph G and a sphere cut decomposition of width $\leq \sqrt{4.5 \cdot |V(G)|}$, we can compute an optimal solution of MDBCS$_d$ in G in $O\left((d+1)^{4.24\sqrt{n}} \cdot 5^{4.24\sqrt{n}} \cdot n^{3/2}\right)$ steps (by Proposition 7.1). The same argument applies to all the variants of the problem discussed above.

In addition, we can derive a subexponential exact algorithm for the following problem on planar graphs: Minimum Degree Spanning Tree (MDST). The MDST problem consists in, given an undirected unweighted graph G, finding a spanning tree of G with minimizes the maximum degree over all the spanning trees of G. This problem has been widely studied in the literature (cf. for instance [130]), and we are unaware of the existence of subexponential exact algorithms on planar graphs. Our algorithm works as follows: given a planar graph G, we find an optimal solution H_d of VMDBCS$_d$ in G for $d = 2, \ldots, n-1$. Let d^* be the first value of d for which $|V(H_d)| = n$. Then an optimal solution of MDST in G is given by any spanning tree of H_{d^*}.

A graph is *supereulerian* if it has a spanning Eulerian subgraph [67]. Combining the ideas of the algorithm above with the ideas of Section 7.6.3 yields a subexponential exact algorithm to decide whether a planar graph is supereulerian or not.

7.7 Conclusions

In this chapter we obtained a $2^{O(\sqrt{k})}n^{O(1)}$ algorithm for k-MDBCS$_d$ and related problems on planar graphs. Several interesting problems remain open. First, it seems natural to try to improve the worst-case running time of our algorithms. Much more challenging is to find subexponential parameterized algorithms for the edge- or node-weighted versions of the problem. Actually, the weighted versions of our parameters remain minor closed (by an easy modification of Lemma 7.2), however the fundamental difference is that the

combinatorial bound of Lemma 7.3 does not hold anymore. On the other hand, the natural extension of this chapter would be to conceive subexponential parameterized algorithms for k-MDBCS$_d$ on other sparse graph classes, like graphs of bounded genus and, more generally, minor-free families of graphs.

Finally, note that the MDBCS$_d$ problem is equivalent to finding a maximum connected subgraph not containing the star $K_{1,d+1}$ as a topological minor. Many classical NP-hard problems can be expressed as finding a maximum subgraph excluding a fixed graph H as a minor (or induced minor, or subgraph, or induced subgraph, or topological minor), hence conceiving a general framework to design subexponential parameterized algorithms for this class of problems would be a celebrated result.

Chapter 8

Dynamic Programming for Graphs on Surfaces

In this chapter we provide a framework for the design of $2^{\mathcal{O}(k)} \cdot n$ step dynamic programming algorithms for surface-embedded graphs on n vertices of branchwidth at most k. Our technique applies to graph problems for which dynamic programming uses tables encoding set partitions. For general graphs, the best known algorithms for such problems run in $2^{\mathcal{O}(k \cdot \log k)} \cdot n$ steps. That way, we considerably extend the class of problems that can be solved by algorithms whose running times have a *single exponential dependence* on branchwidth, and improve the running time of several existing algorithms. Our approach is based on a new type of branch decomposition called *surface cut decomposition*, which generalizes sphere cut decompositions for planar graphs (see page 176), and where dynamic programming should be applied for each particular problem. The construction of such a decomposition uses a new graph-topological tool called *polyhedral decomposition*. The main result is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is *single exponential*. This fact is proved by a detailed analysis of how non-crossing partitions are arranged on surfaces with boundary and uses diverse techniques from topological graph theory and analytic combinatorics.

Keywords: parameterized algorithms, analytic combinatorics, graphs on surfaces, branchwidth, dynamic programming, polyhedral embeddings, symbolic method, non-crossing partitions.

8.1 Introduction

One of the most important parameters in the design and analysis of graph algorithms is the branchwidth of a graph. Branchwidth, together with its twin parameter of treewidth, can be seen as a measure of the topological resemblance of a graph to a tree. Its algorithmic importance has its origins in the celebrated theorem of Courcelle (see e.g. [86]), stating

that graph problems expressible in monadic second-order logic can be solved in $f(\mathbf{bw}) \cdot n$ (here **bw** is the branchwidth[1] and n is the number of vertices of the input graph). Using the parameterized complexity terminology, this implies that a huge number of graph problems are fixed-parameter tractable when parameterized by the branchwidth of their input graph. As the bounds for $f(\mathbf{bw})$ provided by Courcelle's theorem are huge, the design of tailor-made dynamic programming algorithms for specific problems, so that $f(\mathbf{bw})$ is a simple (preferably single exponential) function, became a natural (and unavoidable) ingredient for many papers on algorithms design (see [39, 58, 100, 199]).

Dynamic programming. Dynamic programming is applied in a bottom-up fashion on a rooted branch decomposition of the input graph, that roughly is a way to decompose the graph into a tree structure of edge bipartitions (the formal definition can be found in page 22). Each bipartition defines a separator of the graph called *middle set*, of cardinality bounded by the branchwidth of the input graph. The decomposition is routed in the sense that one of the parts of each bipartition is the "lower part of the middle set", i.e., the so-far processed one. For each graph problem, dynamic programming requires the suitable definition of tables encoding how potential (global) solutions of the problem are restricted in the middle set and the corresponding lower part. The size of these tables reflects the dependence of **bw** in the running time of the dynamic programming. Defining the tables is not always an easy task, as they depend on the particularities of each problem (some typical examples are shown in Section 8.3). In many cases, problems are grouped together according to the similarities in the way to treat them, and usually this leads to distinct upper bounds for the function $f(\mathbf{bw})$. We define the following categories of dynamic programming algorithms (below S denotes a middle set of a branch decomposition):

(A) those where the tables encode a fixed number of *vertex subsets of* S;

(B) those where the tables encode a fixed number of *connected pairings of vertices of* S; and

(C) those where the tables encode a fixed number of *connected packings of* S *into sets*.

In Categories (B) and (C), by the term *connected* for the pairings (resp. packings) we mean that they correspond to a packing of paths (resp. trees) in the lower part of the middle set S. The above classification also induces a classification of graph problems depending on whether they can be solved by some algorithm in some of the above categories[2]. Notice that the problems in Category (A) belong also to Category (B), and problems in Category (B) are also problems in Category (C). Clearly, the size of the tables for problems in Category (A) is a single exponential function of the middle set size. Therefore, for such problems we have that $f(\mathbf{bw}) = 2^{\mathcal{O}(\mathbf{bw})}$. Such problems are, for instance, 3-Coloring, Vertex Cover, Dominating Set, or Independent Set (see Section I.2.4), whose common characteristic is that the certificate of the solution is a set (or a fixed number of

[1]The original statement of Courcelle's theorem used the parameter of treewidth instead of branchwidth. The two parameters are approximately equivalent, in the sense that the one is a constant factor approximation of the other (see page 22).

[2]To facilitate our presentation, we present in Section 8.3 the dynamic programming algorithms for a problem in Category (A) and a problem in Category (C).

sets) of vertices whose choice is not restricted by some global condition. Unfortunately, when connectivity conditions are applied, the tables of the dynamic programming are of size $2^{O(\mathbf{bw}\cdot\log(\mathbf{bw}))}$ or more. This happens because one needs to encode more information on the way a possible solution of the problem is situated in the middle set S, which usually classifies the problems in categories (B) or (C). Typical problems in Category (B) are Longest Path and Hamiltonian Cycle, where pairings correspond to the connected portions of a solution to the lower part of the middle set. Typical problems in Category (C) are Connected Vertex Cover and Maximum Induced Forest[3], where the connected portions of a solution may be identified by sets of arbitrary cardinality. For Category (B), the size of the tables is lower-bounded by the number of perfect matchings of a complete bipartite graph of k vertices, that is by $2^{\Theta(k\cdot\log k)}$. For Category (C), the size of the tables is lower bounded by the k-th Bell number, that is again lower-bounded by $2^{\Theta(k\cdot\log k)}$. In both cases, this implies algorithms where $f(\mathbf{bw}) = 2^{O(\mathbf{bw}\cdot\log \mathbf{bw})}$.

Single-exponentiality: results and techniques. The most desired characteristic of any dynamic programming algorithm is the single exponential dependence on the branchwidth of the input graph (according to the results in [154], this dependence is optimal for many combinatorial problems). Exponential dependence is trivial for problems in Category (A), and may become possible for the other two categories when we take into account the *structural properties* of the input graph. For problems in Category (B), the first step in this direction was done in [102] for planar graphs (see Chapter 7 for extensions of this technique for problems in Category (C)). The idea in [102] is to use a special type of branch decomposition called *sphere cut* decomposition (introduced in [190], see page 176) that can guarantee that the pairings are non-crossing pairings (because of the connectivity demand) around a virtual edge-avoiding cycle (called *noose*) of the plane where G is embedded. This restricts the number of tables corresponding to a middle set of size k by the k-th Catalan number, which is *single-exponential* in k. That way, single-exponential (in **bw**) running times can be designed for problems in Category (B). The same approach was extended for graphs of Euler genus γ in [99]. The idea was to perform a *planarization* of the input graph by splitting the potential solution into at most γ pieces and then applying the sphere cut decomposition technique of [102] to a more general version of the problem where the number of pairings is still bounded by some Catalan number. This made it possible to avoid dealing with the combinatorial structures in surfaces, where the arrangement of the solutions are harder to handle. The same idea was applied in [101] for H-minor free graphs using much more involved Catalan structures, again for problems in Category (B).

Our results. In this chapter, we follow a different approach in order to design single exponential (in **bw**) algorithms for graphs embedded in surfaces. In particular, we deviate significantly from the planarization technique of [99]. Instead, we extend the concept of sphere cut decomposition from planar graphs to surfaces and we exploit directly the combinatorial structure of the potential solutions in the topological surface. Our approach

[3]Notice that the Maximum Induced Forest problem is equivalent to the Feedback Vertex Set problem. We choose this way to present it in order to make more visible its classification into Category (C).

permits us to provide combinatorial bounds for problems in Category (C). Apart from those mentioned above, examples of such problems are Maximum d-Degree-Bounded Connected Subgraph, Maximum d-Degree-Bounded Connected Induced Subgraph and all the variants studied in Chapter 7, Connected Dominating Set, Connected r-Domination, Connected FVS, Maximum Leaf Spanning Tree, Maximum Full-Degree Spanning Tree, Maximum Eulerian Subgraph, Steiner Tree, and Maximum Leaf Tree. As Category (C) includes the problems in Category (B), our results imply all the results in [99], and with running times whose genus dependence is better than the ones in [99], as discussed in Section 8.14.

Our techniques. Our analysis is based on a special type of branch decomposition of embedded graphs with nice topological properties, which we call *surface cut decomposition* (see Section 8.6). Roughly, the middle sets of such a decomposition are situated along a bounded (by the genus γ) set of nooses of the surface with few (again bounded by γ) common points. The construction of such a decomposition is based on the concept of *polyhedral decomposition* introduced in Section 8.4. In Section 8.5, we prove some basic properties of surface cut decompositions that make it possible to bound the sizes of the tables of the dynamic programming. They correspond to the number of non-crossing partitions of vertex sets laying in the boundary of a generic surface. To count these partitions, we use a powerful technique of analytic combinatorics: *singularity analysis* over expressions obtained by the *symbolic method* (for more on this technique, see the monograph of Flajolet and Sedgewick [117]). The symbolic method gives a precise asymptotic enumeration of the number of non-crossing partitions, that yields the single exponentiality of the table size (see Section 8.7). To solve a problem in Category (C), our approach resides on a common preprocessing step that is to construct the *surface cut decomposition* (Algorithm 2 in Section 8.6). Then, what remains is just to run the dynamic programming algorithm on such a surface cut decomposition. The exponential bound on the size of the tables of this dynamic programming algorithm is provided as a result of our analysis in Theorem 8.4 of Section 8.14.

Section 8.2 is devoted to provide all the preliminaries to read the sequel of the chapter.

8.2 Background and Notation

Sections 8.2.1, 8.2.2, 8.2.3, and 8.2.4 contain the basic background and the notation we will use concerning topological surfaces, graphs embedded in surfaces, carving decompositions and clique sums, and the symbolic method and analytic combinatorics, respectively.

8.2.1 Topological surfaces

In this chapter, we consider compact surfaces with boundary homeomorphic to a finite set (possibly empty) of disjoint circles. We denote the number of connected components of the boundary of a surface Σ by $\beta(\Sigma)$. The surface classification theorem asserts that a compact, connected and without boundary surface is determined, up to homeomorphism,

by its Euler characteristic $\chi(\Sigma)$ and by whether it is orientable or not. More precisely, orientable surfaces are obtained by adding $g \geq 0$ *handles* to the sphere $\mathbb{S}^2$, obtaining the g-torus $\mathbb{T}_g$ with Euler characteristic $\chi(\mathbb{T}_g) = 2-2g$, while non-orientable surfaces are obtained by adding $h > 0$ *cross-caps* to the sphere, hence obtaining a non-orientable surface $\mathbb{P}_h$ with Euler characteristic $\chi(\mathbb{P}_h) = 2 - h$. Writing $\overline{\Sigma}$ for the surface (without boundary) obtained from Σ by gluing a disk on each of the $\beta(\Sigma)$ components of the boundary, we obtain the equality $\chi(\overline{\Sigma}) = \beta(\Sigma) + \chi(\Sigma)$. A subset Π of a surface Σ is *surface-separating* if $\Sigma \setminus \Pi$ has at least 2 connected components.

As a conclusion, our surfaces are determined, up to homeomorphism, by their orientability, their Euler characteristic and the number of connected components of their boundary. For computational simplicity, it is convenient to work with the *Euler genus* $\gamma(\Sigma)$ of a surface Σ, which is defined as $\gamma(\Sigma) = 2 - \chi(\Sigma)$.

8.2.2 Graphs embedded in surfaces

Our main reference for graphs on surfaces is the monograph of Mohar and Thomassen [171]. For a graph G we use the notation (G, τ) to denote that τ is an embedding of G in Σ, whenever the surface Σ is clear from the context. An embedding has *vertices*, *edges*, and *faces*, which are 0, 1, and 2 dimensional open sets, and are denoted $V(G)$, $E(G)$, and $F(G)$, respectively. We use $e(G)$ to denote $|E(G)|$. In a *2-cell embedding*, also called *map*, each face is homeomorphic to a disk. The degree $\mathrm{d}(v)$ of a vertex v is the number of edges incident with v, counted with multiplicity (loops are counted twice). An edge of a map has two ends (also called *half-edges*), and either one or two sides, depending on the number of faces which is incident with. A map is *rooted* if an edge and one of its half-edges and sides are distinguished as the root-edge, root-end and root-side, respectively. Notice that the rooting of maps on orientable surfaces usually omits the choice of a root-side because the underlying surface is oriented and maps are considered up to orientation preserving homeomorphism. Our choice of a root-side is equivalent in the orientable case to the choice of an orientation of the surface. The root-end and -sides define the root-vertex and -face, respectively. The rooted maps are considered up to cell-preserving homeomorphisms preserving the root-edge, -end, and -side. In figures, the root-edge is indicated as an oriented edge pointing away from the root-end and crossed by an arrow pointing towards the root-side (this last, provides the orientation in the surface).

For a graph G, the *Euler genus* of G, denoted $\gamma(G)$, is the smallest Euler genus among all surfaces in which G can be embedded. Determining the Euler genus of a graph is an NP-hard problem [200], hence we assume throughout the paper that we are given an already embedded graph. An *O-arc* is a subset of Σ homeomorphic to $\mathbb{S}^1$. A subset of Σ meeting the drawing only at vertices of G is called *G-normal*. If an O-arc is G-normal, then we call it a *noose*. The *length* of a noose is the number of its vertices. Many results in topological graph theory rely on the concept of *representativity* [184, 190], also called *facewidth*, which is a parameter that quantifies local planarity and density of embeddings. The representativity $\mathbf{rep}(G, \tau)$ of a graph embedding (G, τ) is the smallest length of a non-contractible (i.e., non null-homotopic) noose in Σ. We call an embedding (G, τ) *polyhedral* [171] if G is 3-connected and $\mathbf{rep}(G, \tau) \geq 3$, or if G is a clique and $1 \leq |V(G)| \leq 3$. With abuse of notation, we also say in that case that the graph G itself is polyhedral.

For a given embedding (G,τ), we denote by (G^*,τ) its dual embedding. Thus G^* is the geometric dual of G. Each vertex v (resp. face r) in (G,τ) corresponds to some face v^* (resp. vertex r^*) in (G^*,τ). Also, given a set $S \subseteq E(G)$, we denote as S^* the set of the duals of the edges in S. Let (G,τ) be an embedding and let (G^*,τ) be its dual. We define the *radial graph embedding* (R_G,τ) of (G,τ) (also known as *vertex-face graph embedding*) as follows: R_G is an embedded bipartite graph with vertex set $V(R_G) = V(G) \cup V(G^*)$. For each pair $e = \{v,u\}$, $e^* = \{u^*,v^*\}$ of dual edges in G and G^*, R_G contains edges $\{v,v^*\}$, $\{v^*,u\}$, $\{u,u^*\}$, and $\{u^*,v\}$. Mohar and Thomassen [171] proved that, if $|V(G)| \geq 4$, the following conditions are equivalent: *(i)* (G,τ) is a polyhedral embedding; *(ii)* (G^*,τ) is a polyhedral embedding; and *(iii)* (R_G,τ) has no multiple edges and every 4-cycle of R_G is the border of some face. The *medial graph embedding* (M_G,τ) of (G,τ) is the dual embedding of the radial embedding (R_G,τ) of (G,τ). Note that (M_G,τ) is a Σ-embedded 4-regular graph.

8.2.3 Carving decompositions and clique sums

Recall the definition of branch decompositions and branchwidth from Section I.1.1 (page 22). A *carving decomposition* (T,μ) is similar to a branch decomposition, only with the difference that μ is a bijection between the leaves of the tree and the vertex set of the graph G. For an edge e of T, the counterpart of the middle set, called the *cut set* $\mathbf{cut}(e)$, contains the edges of G with endvertices in the leaves of both subtrees. The counterpart of branchwidth is *carvingwidth*, and is denoted by $\mathbf{cw}(G)$.

Let $G = (V,E)$ be a connected graph. For $S \subseteq V$, we denote by $\delta(S)$ the set of all edges with an end in S and an end in $V \setminus S$. Let $\{V_1,V_2\}$ be a partition of V. If $G[V \setminus V_1]$ and $G[V \setminus V_2]$ are both non-null and connected, we call $\delta(V_1)$ a *bond* of G [190]. In a *bond carving decomposition*, every cut set is a bond of the graph. That is, in a bond carving decomposition, every cut set separates the graph into two connected components.

We now recall the definition of clique sum (see page 162). Let G_1 and G_2 be graphs with disjoint vertex-sets and let $k \geq 0$ be an integer. For $i = 1,2$, let $W_i \subseteq V(G_i)$ form a clique of size k and let G_i' $(i = 1,2)$ be obtained from G_i by deleting some (possibly no) edges from $G_i[W_i]$ with both endpoints in W_i. Consider a bijection $h : W_1 \to W_2$. We define a *clique sum* G of G_1 and G_2, denoted by $G = G_1 \oplus_k G_2$, to be the graph obtained from the union of G_1' and G_2' by identifying w with $h(w)$ for all $w \in W_1$. The integer k is called the *size* of the clique sum.

8.2.4 The symbolic method and analytic combinatorics

The main reference in enumerative combinatorics is the monograph of Flajolet and Sedgewick [117]. The framework introduced in this book gives a language to translate combinatorial conditions between combinatorial classes into equations. This is what is called the *symbolic method* in combinatorics. Later, we can treat these equations as relations between analytic functions. This point of view gives us the possibility to use complex analysis techniques to obtain information about the combinatorial classes. This is the origin of the term *analytic combinatorics*.

For a set $\mathcal{A}$ of objects, let $|\cdot|$ be an application from $\mathcal{A}$ to $\mathbb{N}$, which is called the *size*. A pair $(\mathcal{A}, |\cdot|)$ is called a *combinatorial class*. Define the formal power series (called the *generating function* or *GF* associated to the class) $\mathbf{A}(z) = \sum_{a\in\mathcal{A}} z^{|a|} = \sum_{k=0}^{\infty} a_k z^k$. The constructions we use in this work and their translation into the language of GFs are shown in Table 8.1.

Construction		GF
Union	$\mathcal{A} \cup \mathcal{B}$	$\mathbf{A}(z) + \mathbf{B}(z)$
Product	$\mathcal{A} \times \mathcal{B}$	$\mathbf{A}(z)\mathbf{B}(z)$
Sequence	$\mathbf{Seq}(\mathcal{A})$	$\frac{1}{1-\mathbf{A}(z)}$
Pointing	$\mathcal{A}^\bullet$	$z\frac{\partial}{\partial z}\mathbf{A}(z)$

Table 8.1: Constructions and translations into GF.

The union $\mathcal{A} \cup \mathcal{B}$ of $\mathcal{A}$ and $\mathcal{B}$ refers to the disjoint union of the classes. The cartesian product $\mathrm{A} \times \mathcal{B}$ of $\mathcal{A}$ and $\mathcal{B}$ is the set $\{(a, b) : a \in \mathcal{A}, b \in \mathcal{B}\}$. The sequence $\mathbf{Seq}(\mathcal{A})$ of a set $\mathcal{A}$ corresponds to the set $\mathcal{E} \cup \mathcal{A} \cup \mathcal{A}\times\mathcal{A} \cup \mathcal{A}\times\mathcal{A}\times\mathcal{A} \cup \ldots$. At last, the pointing operator $\mathcal{A}^\bullet$ of a set $\mathcal{A}$ consists in pointing one of the atoms of each element $a \in \mathcal{A}$.

The study of the growth of the coefficients of GFs can be obtained by considering GFs as complex functions which are analytic around $z = 0$. This is the philosophy of analytic combinatorics. The growth behavior of the coefficients depends only on the smallest positive singularity of the GF. Its *location* provides the *exponential growth* of the coefficients, and its *behavior* gives the *subexponential growth* of the coefficients.

The basic results in this area are the so-called *Transfer Theorems for singularity analysis*. These results allows us to deduce asymptotic estimates of an analytic function using its asymptotic expansion near its dominant singularity. The precise statement is claimed in [117]. We use the following reduced version of the theorem (without taking care of technical conditions, which are satisfied in all cases): let $\mathbf{F}(z)$ be a GF with positive coefficients, such that ρ is its smallest real singularity. Suppose that $\mathbf{F}(z)$ admits a singular expansion around $z = \rho$ of the form $\mathbf{F}(z) =_{z\to\rho} C(1 - z/\rho)^{-\alpha} + O\left((1 - z/\rho)^{-\alpha+1/2}\right)$, where C is a constant. Then the Transfer Theorem for singularity analysis states that, for k big enough, the following estimate holds

$$[z^k]\mathbf{F}(z) =_{k\to\infty} C\frac{k^{\alpha-1}}{\Gamma(\alpha)}\rho^{-k}\left(1 + O\left(k^{-1/2}\right)\right). \tag{8.1}$$

8.3 Examples of Dynamic Programming Algorithms

In this section we present two examples of typical dynamic programming algorithms on graphs of bounded branchwidth. The first algorithm solves the Vertex Cover problem and belongs in Category (A) while he second solves the Connected Vertex Cover problem and belongs in Category (C) but not in (B) (nor in (A)).

The standard dynamic programming approach on branch decompositions requires the so-called *rooted* branch decomposition defined as a triple (T, μ, e_r) where (T, μ) is a branch-decomposition of G where T is a tree rooted on a leaf v_l incident to some edge e_r of T. We insist that no edge of G is assigned to v_l and thus $\mathbf{mid}(e_r) = \emptyset$ (for this, we take any edge of a branch decomposition, subdivide it and then connect by e_r the subdivision vertex with a new leaf t_l). The edges of T can be oriented towards the root e_r and for each edge $e \in E(T)$ we denote by E_e the edges of G that are mapped to leaves of T that are descendants of e. We also set $G_e = G[E_e]$ and we denote by $L(T)$ the edges of T that are incident to leaves

of T. Given an edge e heading at a non-leaf vertex v, we denote by $e_1, e_2 \in E(T)$ the two edges with tail v. As both examples below are variants of the vertex cover problem, we can also assume that $|E(T)| = O(k \cdot n)$ as, otherwise, the answer for each problem is trivially negative.

Dynamic programming for Vertex Cover**.** Let G be a graph and $X, X' \subseteq V(G)$ where $X \cap X' = \emptyset$. We say that $\mathbf{vc}(G, X, X') \leq k$ if G contains a vertex cover S where $|S| \leq k$ and $X \subseteq S \subseteq V(G) \setminus X'$. Let $\mathcal{R}_e = \{(X, k) \mid \mathbf{vc}(G_e, X, \mathbf{mid}(e) \setminus X) \leq k\}$ and observe that $\mathbf{vc}(G) \leq k$ iff $(\emptyset, k) \in \mathcal{R}_{e_r}$. For each $e \in E(T)$ we can compute $\mathcal{R}_e$ by using the following dynamic programming formula:

$$\mathcal{R}_e = \begin{cases} \{(X,k) \mid X \neq \emptyset \wedge k \geq |X|\} & \text{if } e \in L(T) \\ \{(X,k) \mid \exists (X_1,k_1) \in \mathcal{R}_{e_1}, \exists (X_2,k_2) \in \mathcal{R}_{e_2} : \\ (X_1 \cup X_2) \cap \mathbf{mid}(e) = X \wedge k_1 + k_2 - |X_1 \cap X_2| \leq k\} & \text{if } e \notin L(T) \end{cases}$$

Notice that for each $e \in E(T)$, $|\mathcal{R}_e| \leq 2^{|\mathbf{mid}(e)|} \cdot k$. Therefore, the above algorithm can check whether $\mathbf{vc}(G) \leq k$ in $O(2^{\mathbf{bw}(G)} \cdot k \cdot |V(T)|)$ steps. Clearly, this simple algorithm is single exponential on $\mathbf{bw}(G)$. Moreover the above dynamic programming machinery can be adapted to many other combinatorial problems where the certificate of the solution is a (non-restricted) subset of vertices (e.g. Dominating Set, 3-Coloring, Independent Set, among others).

Dynamic programming for Connected Vertex Cover**.** Suppose now that we are looking for a *connected* vertex cover of size $\leq k$. Clearly, the above dynamic programming formula does not work for this variant as we should book-keep more information on X towards encoding the connectivity demand.

Let G be a graph, $X \subseteq V(G)$ and $\mathcal{H}$ be a (possibly empty) hypergraph whose vertex set is a subset of X, whose hyperedges are non-empty, pairwise non-intersecting, and such that each vertex of $\mathcal{H}$ belongs to some of its hyperedges (we call such a hypergraph *partial packing* of X). Suppose that $\mathcal{H}$ is a partial packing on $\mathbf{mid}(e)$. We say that $\mathbf{cvc}(G, \mathcal{H}) \leq k$ if G contains a vertex cover S where $|S| \leq k$ and such that if $\mathcal{C}$ is the collection of the connected components of $G_e[S]$, then either $|E(\mathcal{H})| = |\mathcal{C}|$ and $(X, \{X \cap V(C) \mid C \in \mathcal{C}\}) = \mathcal{H}$ or $E(\mathcal{H}) = \emptyset$ and $|\mathcal{C}| = 1$.

As before, let $\mathcal{Q}_e = \{(\mathcal{H}, k) \mid \mathbf{cvc}(G, \mathcal{H}) \leq k\}$ and observe that $\mathbf{cvc}(G) \leq k$ iff $((\emptyset, \emptyset), k) \in \mathcal{Q}_{e_r}$. The dynamic programming formula for computing $\mathcal{Q}_e$ for each $e \in E(T)$ is the following.

$$\mathcal{Q}_e = \begin{cases} \{(\mathcal{H},k) \mid \min\{k, |E(\mathcal{H})|+1\} \geq |V(\mathcal{H})| \geq 1 & \text{if } e \in L(T) \\ \{(\mathcal{H},k) \mid \exists (\mathcal{H}_1,k_1) \in \mathcal{Q}_{e_1}, \exists (\mathcal{H}_2,k_2) \in \mathcal{Q}_{e_2} : \\ V(\mathcal{H}_1) \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) = V(\mathcal{H}_2) \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)), \\ (\mathcal{H}_1 \oplus \mathcal{H}_2)[\mathbf{mid}(e)] = \mathcal{H}, k_1 + k_2 - |V(\mathcal{H}_1) \cap V(\mathcal{H}_2)| \leq k\}, \\ \text{if } E(\mathcal{H}) = \emptyset \text{ then } |E(\mathcal{H}_1 \oplus \mathcal{H}_2)| = 1, \text{and} \\ \text{if } E(\mathcal{H}) \neq \emptyset \text{ then } |E(H_1 \oplus H_2)| = |E(\mathcal{H})| & \text{if } e \notin L(T). \end{cases}$$

In the above formula, $\mathcal{H}_1 \oplus \mathcal{H}_2$ is the hypergraph with vertex set $V(\mathcal{H}_1) \cup V(\mathcal{H}_2)$ where each of its hyperedges contains the vertices of each of the connected components of $\mathcal{H}_1 \cup \mathcal{H}_2$.

Clearly each $\mathcal{H}$ corresponds to a collection of subsets of X and the number of such collections for a given set $\mathbf{mid}(e)$ of r elements is given by the r-th Bell number of r, denoted by B_r. By taking the straightforward upper bound $|B_r| = 2^{O(r \log r)}$, we have that one can check whether an input graph G has a connected vertex cover of size at most k in $O(2^{\mathbf{bw}(G)\cdot\log(\mathbf{bw}(G))} \cdot k \cdot |V(T)|)$ steps.

As the growth of B_r is not single exponential, we cannot hope for a single exponential (in $\mathbf{bw}(G)$) running time for the above dynamic programming procedure and no algorithm is known for this problem that runs in time that is single exponential in $\mathbf{bw}(G)$. The same problem appears for numerous other problems where further restrictions apply to their solution certificates. Such problems can be connected variants of problems in Category (A) and others such as Maximum Induced Forest, Maximum d-Degree-Bounded Connected Subgraph, Maximum d-Degree-Bounded Connected Induced Subgraph and all the variants studied in Chapter 7, Connected Dominating Set, Connected r-Domination, Connected FVS, Maximum Leaf Spanning Tree, Maximum Full-Degree Spanning Tree, Maximum Eulerian Subgraph, Steiner Tree, or Maximum Leaf Tree.

8.4 Polyhedral Decompositions

We introduce in this section *polyhedral decompositions* of graphs embedded in surfaces. Let G be an embedded graph, and let N be a noose in the surface. Similarly to [63], we use the notation $G \rtimes N$ for the graph obtained by cutting G along the noose N and gluing a disk on the obtained boundaries.

Definition 8.1 *Given a graph $G = (V, E)$ embedded in a surface of Euler genus γ, a* polyhedral decomposition *of G is a set of graphs $\mathcal{G} = \{H_1, \ldots, H_\ell\}$ together with a set of vertices $A \subseteq V$ such that*

- $|A| = O(\gamma)$;
- *H_i is a minor of $G[V \setminus A]$, for $i = 1, \ldots, \ell$;*
- *H_i has a polyhedral embedding in a surface of Euler genus at most γ, for $i = 1, \ldots, \ell$; and*
- *$G[V \setminus A]$ can be constructed by joining the graphs of $\mathcal{G}$ applying clique sums of size $0, 1,$ or 2.*

Remark 8.1 *Note that an embedded graph H is not polyhedral if and only if there exists a noose N of length at most 2 in the surface in which H is embedded, such that either N is non-contractible or $V(H) \cap N$ separates H. Indeed, if H has representativity at most 2, then there exists a non-contractible noose N of length at most 2. Otherwise, since H is not polyhedral, H has a minimal separator S of size at most 2. It is then easy to see that there exists a noose containing only vertices of S.*

Algorithm 1 Construction of a polyhedral decomposition of an embedded graph G

Input: A graph G embedded in a surface of Euler genus γ.
Output: A polyhedral decomposition of G.

$A = \emptyset$, $\mathcal{G} = \{G\}$ (the elements in $\mathcal{G}$, which are embedded graphs, are called *components*).
while $\mathcal{G}$ contains a non-polyhedral component H **do**
  Let N be a noose as described in Observation 8.1 in the surface in which H is embedded, and let $S = V(H) \cap N$.
  if N is non-surface-separating **then**
    Add S to A, and replace in $\mathcal{G}$ component H with $H[V(H) \setminus S] \rtimes N$.
  end if
  if N is surface-separating **then**
    Let H_1, H_2 be the subgraphs of $H \rtimes N$ corresponding to the two surfaces occurring after splitting H
    if $S = \{u\} \cup \{v\}$ and $\{u, v\} \notin E(H)$ **then**
      Add the edge $\{u, v\}$ to H_i, $i = 1, 2$.
    end if
    Replace in $\mathcal{G}$ component H with the components of $H \rtimes N$ containing at least one edge of H.
  end if
end while
return $\{\mathcal{G}, A\}$.

Algorithm 1 provides an efficient way to construct a polyhedral decomposition, as it is stated in Proposition 8.1.

In the above algorithm, the addition of an edge $\{u, v\}$ represents the existence of a path in G between u and v that is not contained in the current component.

Proposition 8.1 *Given a graph G on n vertices embedded in a surface, Algorithm 1 constructs a polyhedral decomposition of G in $O(n^3)$ steps.*

Proof: We first prove that the the output $\{\mathcal{G}, A\}$ of Algorithm 1 is indeed a polyhedral decomposition of G, and then we analyze the running time.

Let us see that each component of $\mathcal{G}$ is a minor of $G[V \setminus A]$. Indeed, the only edges added to G by Algorithm 1 are those between two non-adjacent vertices u, v that separate a component H into several components $H_1, \ldots, H_\ell$. For each component H_i, $i = 1, \ldots, \ell$, there exists a path between u and v in $H \setminus H_i$ (provided that the separators of size 1 have been already removed, which can we assumed w.l.o.g.), and therefore the graph obtained from H_i by adding the edge $\{u, v\}$ is a minor of H, which is inductively a minor of $G[V \setminus A]$. Also, each component of $\mathcal{G}$ is polyhedral by definition of the algorithm.

As a non-separating noose is necessarily non-contractible, each time some vertices are moved to A the Euler genus of the surfaces strictly decreases [171, Lemma 4.2.4]. Therefore, $|A| = O(\gamma)$.

By the construction of the algorithm, it is also clear that each component of $\mathcal{G}$ has a polyhedral embedding in a surface of Euler genus at most γ. Finally, $G[V \setminus A]$ can be constructed by joining the graphs of $\mathcal{G}$ applying clique sums of size $0, 1$, or 2.

Thus, $\{\mathcal{G}, A\}$ is a polyhedral decomposition of G by Definition 8.1.

We now analyze the running time of the algorithm. Separators of size at most 2 can be found in $O(n^2)$ steps [148]. A noose with respect to a graph H corresponds to a cycle in the radial graph of H, hence can also be found[4] in $O(n^2)$ (using that the number of edges of a bounded-genus graph is linearly bounded by its number of vertices). Since each time that we find a *small* separator we decrease the size of the components, the running time of the algorithm is $O(n^3)$. □

8.5 Some Topological Lemmata

In this section we state some topological lemmata that are used in Sections 8.6 and 8.7. In particular, Lemmata 8.1 and 8.2 are used in the proof of Theorem 8.1 (page 197) while Lemma 8.3 is used in the proof of Theorem 8.2 (page 204).

Given a graph G embedded in a surface of Euler genus γ, its dual G^* and a spanning tree C^* of G^*, we call $C = \{e \in E(G) \mid e^* \in E(C^*)\}$ a *spanning cotree* of G. We define a *tree-cotree partition* (cf. [109]) of an embedded graph G to be a triple (T, C, X) where T is a spanning tree of G, C is a spanning cotree of G, $X \subseteq E(G)$, and the three sets $E(T)$, C, and X form a partition of $E(G)$. Eppstein proved [109, Lemma 3.1] that if T and C^* are forests such that $E(T)$ and C are disjoint, we can make T become part of a spanning tree T' and C become part of a spanning cotree disjoint from T', extending T and C to a tree-cotree decomposition. We can now announce the following lemma from [109, Lemma 3.2].

Lemma 8.1 (Eppstein [109]) *If (T, C, X) is a tree-cotree decomposition of a graph G embedded in a surface of Euler genus γ, then $|X| = O(\gamma)$.*

Lemma 8.2 *Let Σ be a surface without boundary. Let S be a set of (not necessary disjoint) $O(\gamma(\Sigma))$ cycles such that $\Sigma \backslash S$ has 2 connected components. Let H be the graph corresponding to the union of the cycles in S. Then $\sum_{v \in V(H)}(\mathrm{d}(v) - 2) = O(\gamma(\Sigma))$.*

Proof: Let κ be the number of cycles, so by assumption $\kappa = O(\gamma(\sigma))$. The first step consists in simplifying the problem. Let H' be the graph obtained from H by dissolving vertices of degree 2 (except from the case of isolated cycles, where we obtain a single vertex of degree 2 and a loop). Each dissolved vertex had degree 2, so the sum $\sum_{v \in V(H')}(\mathrm{d}(v) - 2)$ coincides with $\sum_{v \in V(H)}(\mathrm{d}(v) - 2)$. Additionally, by assumption H' separates Σ into 2 connected components Σ' and Σ''. Let $H'_1, H'_2, \ldots, H'_r$ be the maximal connected subgraphs of H'. In particular, $r \leq \kappa$.

Some of these connected subgraphs may be incident with Σ' but not with Σ'', or conversely. Additionally, there is at least one connected subgraph H'_i incident with both connected components. W.l.o.g. we assume that the subgraphs $H'_1, H'_2, \ldots, H'_p$ are incident only with Σ', $H'_{p+1}, \ldots, H'_q$ are incident with both components, and $H'_{q+1}, \ldots, H'_r$ are incident only with Σ''. It is obvious that there exists a path joining a vertex of H'_i with a vertex of H'_{i+1} if $1 \leq i \leq q-1$ or $p+1 \leq i \leq r-1$.

[4] A shortest non-contractible cycle can be found in $O(2^{O(\gamma \log \gamma)} n^{4/3})$ steps [63]. This time improves on $O(n^3)$ for a big range of values of γ.

From graphs $H'_1, H'_2, \ldots, H'_p, \ldots, H'_q$ (the ones which are incident with Σ') we construct a new graph G_1 in the following inductive way: we start taking H'_q and H'_{q-1}, and a path joining a vertex of H'_q to a vertex of H'_{q-1}. This path exists because H'_q and H'_{q-1} are incident with Σ'. Consider the graph obtained from H'_q and H'_{q-1} by adding an edge that joins this pair of vertices. Then, delete H'_q and H'_{q-1} from the initial list and add this new connected graph. This procedure is done $q-1$ times. At the end, we obtain a connected graph G' incident with both Σ' and Σ'' where each vertex has degree at least 3. Finally, we apply the same procedure with $G', H'_{q+1}, \ldots, H'_r$, obtaining a connected graph G. Observe also that

$$\sum_{v \in V(H)} (\mathrm{d}(v) - 2) \leq \sum_{v \in V(G)} (\mathrm{d}(v) - 2) < \sum_{v \in V(G)} \mathrm{d}(v) = 2|E(G)|.$$

In what follows, we obtain upper bounds for $2|E(G)|$. Observe that H' defines a pair of faces over Σ, not necessarily disks. In the previous construction of G, every time we add an edge either we subdivide a face into two parts or not. Consequently, the number of faces that G defines over Σ is at most $2+\kappa$. The next step consists in reducing the surface in the following way: let f be a face determined by G over Σ. If f is contractible, we do nothing. If not, there is a non-contractible cycle $\mathbb{S}^1$ contained on f. Consider the connected component of $\Sigma \rightarrowtail \mathbb{S}^1$ which contains G (call it Σ_1). It is obvious that G defines a decomposition of Σ_1, $\gamma(\Sigma_1) \leq \gamma(\Sigma)$, and the number of faces has been increased by at most one. Observe that for each operation $\rightarrowtail$ we reduce the Euler genus and we create at most one face. As the Euler genus is finite, the number of $\rightarrowtail$ operations is also finite. This gives rise to a surface Σ_s, where $s \leq \gamma(\Sigma)$, such that all faces determined by G are contractible. Additionally, the number of faces that G determines over Σ_s is smaller than $2 + \kappa + \gamma(\Sigma)$.

G defines a map on Σ_s (i.e., all faces are contractible), and consequently we can apply Euler's formula. Then $|F(G)| + |V(G)| = |E(G)| + 2 - \gamma(\Sigma_s)$. Then, $|F(G)| \leq 2 + \kappa + \gamma(\Sigma)$, so $|E(G)| + 2 - \gamma(\Sigma_s) = |V(G)| + |F(G)| \leq |V(G)| + 2 + \kappa + \gamma(\Sigma)$. The degree of each vertex is at least 3, thus $3|V(G)| \leq 2|E(G)|$. Substituting this condition in the previous equation, we obtain

$$|E(G)| + 2 - \gamma(\Sigma_s) \leq |V(G)| + 2 + \kappa + \gamma(\Sigma) \leq \frac{2}{3}|E(G)| + 2 + \kappa + \gamma(\Sigma).$$

Isolating $|E(G)|$, we get that $2|E(G)| \leq 6\kappa + 6\gamma(\Sigma_s) + 6\gamma(\Sigma) \leq 6\kappa + 12\gamma(\Sigma)$. This bound immediately translates into the statement of Lemma 8.2. □

We need another topological result, which deals with cycles that separate a given surface, and whose proof is an immediate consequence of [171, Proposition 4.2.1].

Lemma 8.3 *Let Σ be a surface with boundary and let $\mathbb{S}^1$ be a separating cycle. Let V_1 and V_2 be the connected components of $\Sigma \rightarrowtail \mathbb{S}^1$. Then $\gamma(\Sigma) = \gamma(V_1) + \gamma(V_2)$.*

8.6 Surface Cut Decompositions

Sphere cut decompositions [190] (see page 176) have proved to be very useful to analyze the running time of algorithms based on dynamic programming over branch decompositions on planar graphs [C20,100–102]. In this section we generalize sphere cut decompositions to

graphs on surfaces; we call them *surface cut decompositions*. First we need a topological definition. A subset Π of a surface Σ is *fat-connected* if for every two points $p, q \in \Pi$, there exists a path $P \subseteq \Pi$ such that for every $x \in P$, $x \neq p, q$, there exists a subset D homeomorphic to an open disk such that $x \in D \subseteq \Pi$. We can now define the notion of surface cut decomposition.

Definition 8.2 *Given a graph G embedded in a surface Σ, a* surface cut decomposition *of G is a branch decomposition (T, μ) of G such that, for each edge $e \in E(T)$, there is a subset of vertices $A_e \subseteq V(G)$ with $|A_e| = O(\gamma(\Sigma))$ and either*

- $|\mathbf{mid}(e) \setminus A_e| \leq 2$, *or*
- *there exists a polyhedral decomposition $\{\mathcal{G}, A\}$ of G and a graph $H \in \mathcal{G}$ such that*
  - $A_e \subseteq A$;
  - $\mathbf{mid}(e) \setminus A_e \subseteq V(H)$;
  - *the vertices in $\mathbf{mid}(e) \setminus A_e$ are contained in a set $\mathcal{N}$ of $O(\gamma(\Sigma))$ nooses, such that the total number of occurrences in $\mathcal{N}$ of the vertices in $\mathbf{mid}(e) \setminus A_e$ is $|\mathbf{mid}(e) \setminus A_e| + O(\gamma(\Sigma))$; and*
  - $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ *contains exactly two connected components, which are both fat-connected.*

Note that a sphere cut decomposition is a particular case of a surface cut decomposition when $\gamma = 0$, by taking $A_e = \emptyset$, $\mathcal{G}$ containing only the graph itself, and all the vertices of each middle set contained in a single noose.

We need some definitions and auxiliary results to be applied for building surface cut decompositions. In the same spirit of [127, Theorem 1] we can prove the following lemma. We omit the proof here since the details are very similar[5] to the proof in [127].

Lemma 8.4 *Let (G, τ) and (G^*, τ) be dual polyhedral embeddings in a surface of Euler genus γ and let (M_G, τ) be the medial graph embedding. Then* $\max\{\mathbf{bw}(G), \mathbf{bw}(G^*)\} \leq \mathbf{cw}(M_G)/2 \leq 6 \cdot \mathbf{bw}(G) + 2\gamma + O(1)$. *In addition, given a branch decomposition of G of width at most k, a carving decomposition of M_G of width at most $12k$ can be found in linear time.*

Lemma 8.5 (folklore) *The removal of a vertex from a non-acyclic graph decreases its branchwidth by at most* 1.

Lemma 8.6 *Let G be a graph and let $\mathcal{G}$ be a collection of graphs such that G can be constructed by joining graphs in $\mathcal{G}$ applying clique sums of size 0, 1, or 2. Given branch decompositions $\{(T_H, \mu_H) \mid H \in \mathcal{G})\}$, we can compute in linear time a branch decomposition (T, μ) of G such that $\mathbf{w}(T, \mu) \leq \max\{2, \{\mathbf{w}(T_H, \mu_H) \mid H \in \mathcal{G}\}\}$. In particular, $\mathbf{bw}(G) \leq \max\{2, \{\mathbf{bw}(H) \mid H \in \mathcal{G}\}\}$.*

Proof: Note that if G_1 and G_2 are graphs with no vertex (resp. a vertex, an edge) in common, then $G_1 \cup G_2 = G_1 \oplus_0 G_2$ (resp. $G_1 \oplus_1 G_2$, $G_1 \oplus_2 G_2$). To prove Lemma 8.6, we need the following two lemmata.

[5]The improvement in the multiplicative factor of the Euler genus is obtained by applying more carefully Euler's formula in the proof analogous to that of [127, Lemma 2].

Lemma 8.7 *Let G_1 and G_2 be graphs with at most one vertex in common. Then* $\mathbf{bw}(G_1 \cup G_2) = \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2)\}$.

Proof: Assume first that G_1 and G_2 share one vertex v. Clearly $\mathbf{bw}(G_1 \cup G_2) \geq \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2)\}$. Conversely, for $i = 1, 2$, let (T_i, μ_i) be a branch decomposition of G_i such that $\mathbf{w}(T_i, \mu_i) \leq k$. For $i = 1, 2$, let T_i^v be the minimal subtree of T_i containing all the leaves u_i of T_i such that v is an endpoint of $\mu_i(u_i)$. For $i = 1, 2$, we take an arbitrary edge $\{a_i, b_i\}$ of T_i^v, we subdivide it by adding a new vertex w_i, and then we build a tree T from T_1 and T_2 by adding the edge $\{w_1, w_2\}$. We claim that $(T, \mu_1 \cup \mu_2)$ is a branch decomposition of $G_1 \cup G_2$ of width at most k. Indeed, let us compare the middle sets of $(T, \mu_1 \cup \mu_2)$ to those of (T_1, μ_1) and (T_2, μ_2). First, it is clear that the vertices of $V(G_1) \cup V(G_2) - \{v\}$ appear in $(T, \mu_1 \cup \mu_2)$ in the same middle sets as in (T_1, μ_1) and (T_2, μ_2). Secondly, $\mathbf{mid}(\{w_1, w_2\}) = \{v\}$, since v is a cut-vertex of $G_1 \cup G_2$. Also, for $i = 1, 2$, $\mathbf{mid}(\{a_i, w_i\}) = \mathbf{mid}(\{w_i, b_i\}) = \mathbf{mid}(\{a_i, b_i\})$, and the latter has size at most k as $\mathbf{w}(T_i, \mu_i) \leq k$. For all other edges e of T_i, $i = 1, 2$, $\mathbf{mid}(e)$ is exactly the same in T and in T_i, since if $e \in E(T_i^v)$ then $v \in \mathbf{mid}(e)$ in both T and T_i, and if $e \in E(T_i \setminus T_i^v)$ then $v \notin \mathbf{mid}(e)$ in both T and T_i.

If G_1 and G_2 share no vertices, we can merge two branch decompositions (T_1, μ_1) and (T_2, μ_2) by subdividing a pair of arbitrary edges, without increasing the width. □

Lemma 8.8 ([126]) *Let G_1 and G_2 be graphs with one edge f in common. Then* $\mathbf{bw}(G_1 \cup G_2) \leq \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2), 2\}$. *Moreover, if both endpoints of f have degree at least 2 in at least one of the graphs, then* $\mathbf{bw}(G_1 \cup G_2) = \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2)\}$.

It remains only to show how to merge the branch decompositions (T_1, μ_1), (T_2, μ_2) of two graphs H_1, H_2 in $\mathcal{G}$. We distinguish four cases:

(a) H_1 and H_2 share two vertices v_1, v_2, and the edge $e = \{v_1, v_2\} \in E(G)$. We take the leaves in T_1 and T_2 corresponding to e, we identify them, and we add a new edge whose leave corresponds to e (see Figure 8.1(a)).

(b) H_1 and H_2 share two vertices v_1, v_2, and the edge $e = \{v_1, v_2\} \notin E(G)$. We take the leaves in T_1 and T_2 corresponding to e, we identify them, and we dissolve the common vertex (see Figure 8.1(b)).

(c) H_1 and H_2 share one vertex v. We take two edges b, c in T_1, T_2 whose leaves correspond to edges containing v, we subdivide them and add a new edge between the newly created vertices (see Figure 8.1(c)).

(d) H_1 and H_2 share no vertices. We do the construction of case (c) for any two edges of the two branch decompositions.

The above construction does not increase the branchwidth by Lemmata 8.7 and 8.8. □

Given an embedded graph G and a carving decomposition (T, μ) of its medial graph M_G, we define a *radial decomposition* (T^*, μ^*) of the dual graph R_G, where $T^* = T$ and μ^* is a bijection from the leaves of T to the set of faces of R_G defined as follows: for each edge

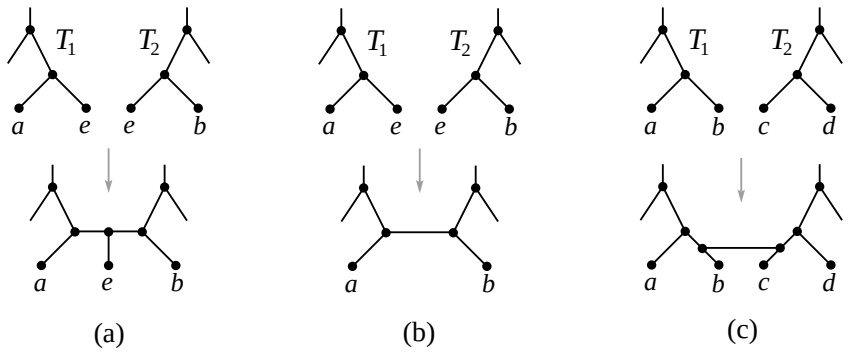

Figure 8.1: Merging branch decompositions (T_1, μ_1) and (T_2, μ_2) of two components H_1 and H_2 in a polyhedral decomposition $\{\mathcal{G}, A\}$ of $G = (V, E)$. There are three cases: (a) H_1 and H_2 share two vertices v_1, v_2 and the edge $e = \{v_1, v_2\}$ is in E; (b) H_1 and H_2 share two vertices v_1, v_2 and $e = \{v_1, v_2\}$ is *not* in E; (c) H_1 and H_2 share one vertex v.

$e \in E(T)$, $\mu^*(e) = f$, where f is the face in R_G corresponding to the vertex $u_f \in V(M_G)$ such that $\mu(e) = u_f$. Each edge $e \in E(T^*)$ partitions the faces of R_G into two sets F_1 and F_2. We define the *border set* of e, denoted $\mathbf{bor}(e)$, as the set of edges of R_G that belong to both F_1 and F_2. Note that F_1 and F_2 may intersect also in vertices, not only in edges. If (T, μ) is a bond carving decomposition of M_G, then the associated radial decomposition (also called *bond*) has nice connectivity properties. Indeed, in a bond carving decomposition, every cut set partitions the vertices of M_G into two subsets V_1, V_2 such that both $M_G[V_1]$ and $M_G[V_2]$ are non-null and connected. This property, seen in the radial decomposition of R_G, implies that each edge $e \in E(T^*)$ corresponds to a partition of the faces of R_G into two sets F_1 and F_2, namely *black* and *white* faces (naturally partitioning the edges into *black*, *white*, and *grey*), such that it is possible to reach any black (resp. white) face from any black (resp. white) face only crossing black (resp. white) edges. In other words, each set of monochromatic faces is fat-connected.

Remark 8.2 *Recall that all the faces of a radial graph R_G are* tiles*, that is, each face has exactly 4 edges. Also, each one of those tiles corresponds to a pair of dual edges e and e^* of G and G^*, respectively. Given a carving decomposition (T, μ) of M_G (or equivalently, a radial decomposition (T^*, μ^*) of R_G), one can obtain in a natural way branch decompositions of G and G^* by redefining the bijection μ from the leaves of T to the edges of G (or G^*) that correspond to the faces of R_G.*

We provide now an algorithm to construct a surface graph decomposition of an embedded graph. The proof of Theorem 8.1 uses Proposition 8.1, topological Lemmata 8.1 and 8.2, and the results of the current section.

Theorem 8.1 *Given a graph G on n vertices embedded in a surface of Euler genus γ, with $\mathbf{bw}(G) \leq k$, Algorithm 2 constructs, in $2^{3k+O(\log k)}n^3$ steps, a surface cut decomposition (T, μ) of G of width at most $27k + O(\gamma)$.*

Proof: We prove, in this order, that (1) the output (T, μ) of Algorithm 2 is indeed a surface cut decomposition of G; (2) the width of (T, μ) is at most $27\mathbf{bw}(G) + O(\gamma)$; and (3) the claimed running time.

Algorithm 2 Construction of a surface cut decomposition of an embedded graph G

Input: An embedded graph G.
Output: A surface cut decomposition of G.

Compute a polyhedral decomposition $\{\mathcal{G}, A\}$ of G, using Algorithm 1.
for each component H of $\mathcal{G}$ **do**
1. Compute a branch decomposition (T'_H, μ'_H) of H, using [36, Theorem 3.8].
2. Transform (T'_H, μ'_H) to a carving decomposition (T^c_H, μ^c_H) of the medial graph M_H, using Lemma 8.4.
3. Transform (T^c_H, μ^c_H) to a *bond* carving decomposition (T^b_H, μ^b_H) of M_H, using [190].
4. Transform (T^b_H, μ^b_H) to a branch decomposition (T_H, μ_H) of H, using Observation 8.2.

end for
Construct a branch decomposition (T, μ) of G by merging, using Lemma 8.6, the branch decompositions $\{(T_H, \mu_H) \mid H \in \mathcal{G}\}$, and by adding the set of vertices A to all the middle sets.
return (T, μ).

(1) (T, μ) is a surface cut decomposition of G.

We shall prove that all the properties of Definition 8.2 are fulfilled. For each $e \in E(T)$ we set $A_e = A \cap \mathbf{mid}(e)$, where A is the set of vertices output by Algorithm 1. Hence, by Proposition 8.1, $|A| = O(\gamma)$.

By construction, it is clear that (T, μ) is a branch decomposition of G. In (T, μ), there are some edges that have been added in the last step of Algorithm 2, in order to merge branch decompositions of the graphs in $\mathcal{G}$, with the help of Lemma 8.6. Let e be such an edge. Since $\{\mathcal{G}, A\}$ is a polyhedral decomposition of G, any pair of graphs in $\mathcal{G}$ share at most 2 vertices, hence $|\mathbf{mid}(e) \setminus A_e| \leq 2$.

All other edges of (T, μ) correspond to an edge of a branch decomposition of some polyhedral component $H \in \mathcal{G}$. Let henceforth e be such an edge. Therefore, $\mathbf{mid}(e) \setminus A_e \subseteq V(H)$. To complete this part of the proof, we prove in a sequence of three claims that the remaining conditions of Definition 8.2 hold.

Claim 8.1 *The vertices in* $\mathbf{mid}(e) \setminus A_e$ *are contained in a set* $\mathcal{N}$ *of* $O(\gamma)$ *nooses.*

Proof: The proof uses the tree-cotree partition defined in Section 8.5.

Recall that e is an edge that corresponds to a branch decomposition (T_H, μ_H) of a polyhedral component H of $\mathcal{G}$. The branch decomposition (T_H, μ_H) of H has been built by Algorithm 2 from a bond carving decomposition of its medial graph M_H, or equivalently from a bond radial decomposition of its radial graph R_H. Due to the fact that the carving decomposition of M_H is bond, edge e partitions the vertices of M_H into two sets – namely, *black* and *white* vertices – each one inducing a connected subgraph of M_H. There are three types of edges: *black*, *white*, and *grey*, according to whether they belong to faces of the same color (black or white) or not. Therefore, the corresponding black and white faces also induce connected subgraphs of R_H, in the sense that it is always possible to reach any black (resp. white) face from any black (resp. white) face only crossing black (resp. white) edges.

Let us now see which is the structure of the subgraph of R_H induced by the edges F belonging to both black and white faces look like. Since each edge of R_H contains a

vertex of H and another from H^*, the vertices in $\mathbf{mid}(e)$ are contained in $R_H[F]$, so it suffices to prove that $R_H[F]$ can be partitioned into a set of $O(\gamma)$ cycles (possibly sharing some vertices).

To this end, first note that in $R_H[F]$ all vertices have even degree. Indeed, let $v \in V(R_H[F])$, and consider a clockwise orientation of the edges incident to v in $R_H[F]$. Each of such edges alternates from a black to a white face, or viceversa, so beginning from an arbitrary edge and visiting all others edges in the clockwise order, we deduce that the number of edges incident to v is necessarily even.

Therefore, $R_H[F]$ can be partitioned into a set of cycles. Let us now bound the number of such cycles. For simplicity, let us identify a pair of dual edges e and e^* as the same edge. Since the subgraph induced by the black (resp. white) faces is connected, we can consider a spanning tree T_B^* (resp. T_W^*) of the black (resp. white) faces. Merge both trees by adding an edge e_0^*, and let T^* be the resulting tree. Let T be a spanning tree of the dual graph disjoint from T^* (such a spanning tree exists by [109, Lemma 3.1]). Now consider the tree-cotree partition (T, T^*, X), where X is the set of edges of R_H that are neither in T nor in T^*.

The edges of T^*, except e_0^*, separate faces of the same color. Therefore, the set $F \in E(R_H)$ of edges separating faces of different color is contained in $T \cup \{e_0\} \cup X$. Since T is a tree, each cycle of $R_H[F]$ uses at least one edge in $\{e_0\} \cup X$. Therefore, $R_H[F]$ can be partitioned into at most $1 + |X|$ cycles. The result follows from the fact that (T, T^*, X) is a tree-cotree partition and therefore $|X| = O(\gamma)$ by Lemma 8.1. □

Claim 8.2 *$\bigcup_{N \in \mathcal{N}} N$ separates Σ into 2 fat-connected components.*

Proof: By Claim 8.1, the vertices in $\mathbf{mid}(e) \setminus A_e$ are contained in $\bigcup_{N \in \mathcal{N}} N$. The claim holds from the fact that for each component H of $\mathcal{G}$, (T_H^b, μ_H^b) is a bond carving decomposition of M_H, and by taking into account the discussion before Observation 8.2 in Section 8.6. □

Claim 8.3 *The total number of occurrences in $\mathcal{N}$ of the vertices in $\mathbf{mid}(e) \setminus A_e$ is $|\mathbf{mid}(e) \setminus A_e| + O(\gamma)$.*

Proof: By Claim 8.2, $\bigcup_{N \in \mathcal{N}} N$ separates Σ into 2 fat-connected components. Let H be the graph induced in Σ by the nooses in $\mathcal{N}$. The claim can then be rephrased as $\sum_{v \in V(H)} (\mathrm{d}(v) - 2) = O(\gamma)$, which holds by Lemma 8.2 in Section 8.5. □

(2) The width of (T, μ) is at most $27 \cdot \mathbf{bw}(G) + O(\gamma)$.

For simplicity, let $k = \mathbf{bw}(G)$. By Proposition 8.1, each polyhedral component H is is a minor of G, hence $\mathbf{bw}(H) \leq k$ for all $H \in \mathcal{G}$. In Step 1 of Algorithm 2, we compute a branch decomposition (T'_H, μ'_H) of H of width at most $k' = \frac{9}{2}k$, using Amir's algorithm [36, Theorem 3.8]. In Step 2, we transform (T'_H, μ'_H) to a carving decomposition (T_H^c, μ_H^c) of the medial graph M_H of H of width at most $12k'$, using Lemma 8.4. In Step 3, we transform (T_H^c, μ_H^c) to a *bond* carving decomposition (T_H^b, μ_H^b) of M_H of width at most $12k'$, using the algorithm of [190]. Then, using

Observation 8.2, we transform in Step 4 (T_H^b, μ_H^b) to a branch decomposition (T_H, μ_H) of H. By the proof of Claim 8.1, the discrepancy between $\mathbf{w}(T_H, \mu_H)$ and $\mathbf{w}(T_H^b, \mu_H^b)/2$ is at most the bound provided by Lemma 8.2, i.e., $O(\gamma)$. Therefore, $\mathbf{w}(T_H, \mu_H) \leq 6k' + O(\gamma) = 27k + O(\gamma)$, for all $H \in \mathcal{G}$.

Finally, we merge the branch decompositions of the polyhedral components to obtain a branch decomposition (T, μ) of G, by adding the vertices in A to all the middle sets. Combining the discussion above with Lemmata 8.5 and 8.6, and using that $|A| = O(\gamma)$, we get that

$$\begin{aligned} \mathbf{w}(T, \mu) &\leq \max\{2, \{\mathbf{w}(T_H, \mu_H) \mid H \in \mathcal{G}\}\} + |A| \\ &\leq 27k + O(\gamma) + |A| \\ &= 27k + O(\gamma). \end{aligned}$$

(3) Algorithm 2 runs in $2^{3k+O(\log k)}n^3$ steps.

We analyze sequentially the running time of each step. First, we compute a polyhedral decomposition of G using Algorithm 1 in $O(n^3)$ steps, by Proposition 8.1. Then, we run Amir's algorithm in each component in Step 1, which takes $O(2^{3k}k^{3/2}n^2)$ steps [36, Theorem 3.8]. Step 2 can be done in linear time by Lemma 8.4. Step 3 can be done in $O(n^2)$ time [190]. Step 4 takes linear time by Observation 8.2. Merging the branch decompositions can clearly be done in linear time. Finally, since any two elements in $\mathcal{G}$ share at most two vertices, the overall running time is the claimed one.

□

How surface cut decompositions are used for dynamic programming. We shall now discuss how surface cut decompositions guarantee good upper bounds on the size of the tables of dynamic programming algorithms for problems in Category (C). The size of the tables depends on how many ways a partial solution can intersect a middle set during the dynamic programming algorithm. The interest of a surface cut decomposition is that the middle sets are placed on the surface in such a way that permits to give a precise asymptotic enumeration of the size of the tables. Indeed, in a surface cut decomposition, once we remove a set of vertices whose size is linearly bounded by γ, the middle sets are either of size at most two (in which case the size of the tables is bounded by a constant) or are situated around a set of $O(\gamma)$ nooses, where vertices can be repeated at most $O(\gamma)$ times. In such a setting, the number of ways that a partial solution can intersect a middle set is bounded by the number of non-crossing partitions of the boundary-vertices in a fat-connected subset of the surface (see Definition 8.2). By splitting the boundary-vertices that belong to more than one noose, we can assume that these nooses are mutually disjoint. That way, we reduce the problem to the enumeration of the non-crossing partitions of $O(\gamma)$ disjoint nooses containing ℓ vertices, which are $2^{O(\ell)} \cdot \ell^{O(\gamma)} \cdot \gamma^{O(\gamma)}$, as we prove in the following section (Theorem 8.3). Observe that the splitting operation increases the size of the middle sets by at most $O(\gamma)$, therefore $\ell = k + O(\gamma)$ and this yields an upper bound of $2^{O(k)} \cdot k^{O(\gamma)} \cdot \gamma^{O(\gamma)}$ on the size of the tables of the dynamic programming. In Section 8.7 we use singularity analysis over expressions obtained by the symbolic method to count the number of such non-crossing partitions. Namely, in Sections 8.7.1 and 8.7.2 we give a

precise estimate of the number of non-crossing partitions in surfaces with boundary. Then we incorporate in Section 8.7.3 two particularities of surface cut decompositions: Firstly, we deal with the set A of vertices originating from the polyhedral decomposition. These vertices are not situated around the nooses that disconnect the surface into two connected components, and this is why they are treated as *apices* in the enumeration. Secondly, we take into account that, in fact, we need to count the number of non-crossing *packings* rather than the number of non-crossing partitions, as a solution may not intersect *all* the vertices of a middle set, but only a subset. The combinatorial results of Section 8.7 are of interest by themselves, as they are a natural extension to higher-genus surfaces of the classical non-crossing partitions in the plane, which are enumerated by the Catalan numbers (see e.g. [116]).

8.7 Non-crossing Partitions in Surfaces with Boundary

In this section we obtain upper bounds for non-crossing partitions in surfaces with boundary. The concept of a non-crossing partition in a general surface is not as simple as in the case of the disk, and must be defined carefully. In Section 8.7.1 we set up our notation. In Section 8.7.2 we obtain a tree-like structure that provides a way to obtain asymptotic estimates. As a main tool we employ map enumeration techniques. To conclude, we extend in Section 8.7.3 the enumeration to two more general families.

8.7.1 2-zone decompositions and non-crossing partitions

Let Σ be a surface with boundary. A 2-*zone decomposition of* Σ is a decomposition of Σ where all vertices lay in the boundary of Σ and there is a coloring of the faces using 2 colors (black and white) such that every vertex is incident (possibly more than once) with a unique black face. Black faces are also called *blocks*. A 2-zone decomposition is *regular* if every block is contractible. All 2-zone decompositions are *rooted*: every connected component of the boundary of Σ is edge-rooted. We denote by $\mathcal{S}_\Sigma(k), \mathcal{R}_\Sigma(k)$ the set of general and regular 2-zone decompositions of Σ with k vertices, respectively. A 2-zone decomposition s over Σ defines a non-crossing partition $\pi_\Sigma(s)$ over the set of vertices. Let $\Pi_\Sigma(k)$ be the set of non-crossing partitions of Σ with k vertices. The main objective in this section consists in obtaining bounds for $|\Pi_\Sigma(k)|$. The critical observation is that each non-crossing partition is defined by a 2-zone decomposition. Consequently, $|\Pi_\Sigma(k)| \leq |\mathcal{S}_\Sigma(k)|$. The strategy to enumerate this second set consists in reducing the enumeration to simpler families of 2-zone decompositions. More specifically, the following proposition shows that it is sufficient to study regular decompositions:

Proposition 8.2 *Let $s \in \mathcal{S}_\Sigma$ be a 2-zone decomposition of Σ and let $\pi_\Sigma(s)$ be the associated non-crossing partition. Then there exists a regular 2-zone decomposition $m \in \mathcal{R}_\Sigma$ such that $\pi_\Sigma(s) = \pi_\Sigma(m)$.*

Proof: First we need a definition and a basic topological lemma. Let Σ_1 and Σ_2 be surfaces with boundary, possibly not connected. We write $\Sigma_2 \subset \Sigma_1$ if there exists a continuous injection $i : \Sigma_2 \hookrightarrow \Sigma_1$ such that $i(\Sigma_2)$ is homeomorphic to Σ_2. If s is a 2-zone decomposition

of Σ_2 and $\Sigma_2 \subset \Sigma_1$, then the injection i induces a 2-zone decomposition $i(s)$ on Σ_1 such that $\pi_{\Sigma_2}(s) = \pi_{\Sigma_1}(i(s))$. In other words, all 2-zone decompositions over Σ_2 can be realized on a surface Σ_1 which contains Σ_2. Consequently, informally speaking, $\Pi_{\Sigma_2}(k) \subseteq \Pi_{\Sigma_1}(k)$ if $\Sigma_2 \subset \Sigma_1$.

Lemma 8.9 *Let m^* be a regular 2-zone decomposition of Σ_1. Let $\Sigma_1 \subset \Sigma$. Then m^* defines a regular 2-zone decomposition m over Σ such that $\pi_{\Sigma_1}(m^*) = \pi_\Sigma(m)$.*

Proof: [of Lemma 8.9] Let $i : \Sigma_1 \hookrightarrow \Sigma$ be the corresponding injective application, and consider $m = i(m^*)$. In particular, a block π of m^* is topologically equivalent to the block $i(\pi)$: i is a homeomorphism between Σ and $i(\Sigma)$. Hence $i(\pi)$ is an open contractible set and m is regular. □

The basic idea to prove Proposition 8.2 is the construction of a finite sequence of 2-zone decompositions $s_0 = s, s_1, \ldots, s_t = m$, such that $\pi_\Sigma(s_0) = \cdots = \pi_\Sigma(s_t)$ and $s_t = m$ is regular. First, consider a non-contractible block f of s. Suppose that the boundary of f consists of more than one connected component. We define the operation of *joining boundaries* as follows: let l be a path that joins a vertex v in one component of the boundary of f with a vertex u in another component. This path exists because f is a face. Consider also a pair of paths l_1, l_2 that joins these two vertices around the initial path l, as illustrated in Figure 8.2. We define the face f' as the one obtained from f by deleting the face defined by l_1 and l_2 which contains l. Let s_1 be the resulting 2-zone decomposition. Observe that the number of connected components of the boundary of f' is the same as for f minus one, and that $\pi_\Sigma(s) = \pi_\Sigma(s_1)$. We can apply this argument over f as many times as the number of components of the boundary of f. At the end, we obtain a 2-zone decomposition s_{p_1} such that $\pi_\Sigma(s) = \pi_\Sigma(s_1) = \cdots = \pi_\Sigma(s_{p_1})$.

Suppose now that the boundary of the block f has one connected component. Additionally, in this block there are vertices on its boundary which are incident with f more than once. Let v be a vertex incident $r > 1$ times with f. In this case we define the operation of *cutting vertices* as follows: consider the intersection of a small neighborhood of v with f, and delete vertex v. This intersection has r connected components. We define a face by pasting v with only one of these components, and disconnecting the others from v (see Figure 8.2). Then the resulting 2-zone decomposition has the same associated non-crossing partition, and v is incident with the corresponding block exactly once.

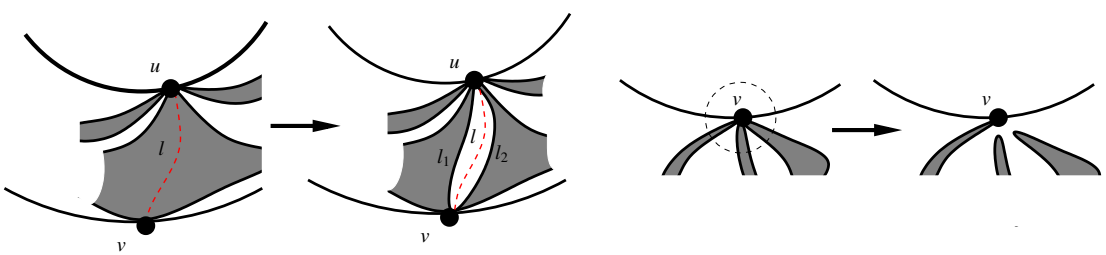

Figure 8.2: The operations of joining boundaries and cutting vertices.

Summarizing, we construct from s a regular 2-zone decomposition in the following way: apply the operation of joining boundaries, and then the operation of cutting vertices. After

this, every block has one boundary and each vertex is incident with its corresponding block exactly once. In this case, a block is either contractible or not. If it is not contractible, let $\mathbb{S}^1$ be a non-contractible cycle, which can be cut using the operator $\succ\!\!\circ$. For all blocks, the number of times we need to apply this operator is bounded by $\gamma(\Sigma)$. At the end, all blocks are contractible and the resulting surface is $\Sigma_1 \subset \Sigma$. So, the resulting 2-zone decomposition m^* is regular, and then by Lemma 8.9 there exists a regular 2-zone decomposition m over Σ such that $\pi_\Sigma(m) = \pi_{\Sigma_1}(m^*) = \pi_\Sigma(s)$, as claimed. □

In other words, $|\Pi_\Sigma(k)| \leq |\mathcal{R}_\Sigma(k)|$ for each value of k. Instead of counting $|\mathcal{R}_\Sigma(k)|$, we reduce our study to the family of regular 2-zone decompositions where each face (block or white face) is contractible. The reason is, as we show later, that this subfamily provides the greatest contribution to the asymptotic enumeration. This set is called the set of *irreducible* 2-zone decompositions of Σ, and it is denoted by $\mathcal{P}_\Sigma(k)$. Equivalently, an irreducible 2-zone decomposition cannot be realized in a proper surface contained in Σ. The details follow.

A generalization of the notion of irreducibility. We shall provide an equivalent definition and an additional property of irreducible 2-zone decompositions. We need the definition stated in the proof of Proposition 8.2 about inclusion of surfaces.

We say that a non-crossing partition π_{Σ_1} is *irreducible* in Σ_1 if there is no realization of π_{Σ_1} in a surface Σ_2 such that $\Sigma_2 \subset \Sigma_1$. This definition is compatible with the notion introduced in Section 8.7.1, as shown in the following lemma:

Lemma 8.10 *Let m be an irreducible 2-zone decomposition of Σ. Then the faces of m are all contractible.*

Proof: We only need to deal with white faces. For a white face whose interior is not an open polygon, there exists a non-contractible cycle $\mathbb{S}^1$. Cutting along $\mathbb{S}^1$ using the operator $\succ\!\!\circ$ we obtain a new surface with boundary Σ' such that $\Sigma' \subset \Sigma$ and m is induced in Σ'. As a conclusion, all faces are contractible. □

8.7.2 Tree-like structures, enumeration, and asymptotic counting

In this subsection we provide estimates for the number of irreducible 2-zone decompositions, which are obtained directly for the surface Σ. This approach is novel and gives upper bounds close to the exact values. The usual technique consists in reducing the enumeration to surfaces of smaller genus, and returning back to the initial one by topological "pasting" arguments. The main point consists in exploiting tree-like structures of the dual graph associated to an irreducible 2-zone decomposition. The main ideas are inspired by [55], where they are used in the context of map enumeration. For simplicity of the presentation, the construction is explained on the disk. The dual graph of a non-crossing partition on the disk is a tree whose internal vertices are bicolored (black color for blocks). An example of this construction is shown in Figure 8.3.

We use this family of trees (and some related ones) in order to obtain a decomposition of elements of the set $\mathcal{P}_\Sigma(k)$. In Section 8.8 the enumeration of this basic family is done,

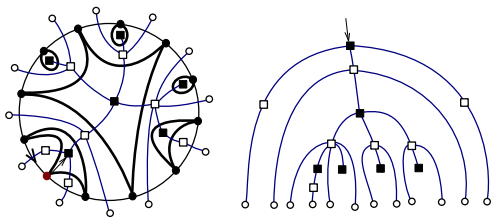

Figure 8.3: A non-crossing partition tree.

as well as the enumeration of the related families. The construction for general surfaces is a generalization of the previous one. An example is shown in the leftmost picture of Figure 8.4. For an element $m \in \mathcal{P}_\Sigma(k)$, denote by M the resulting map on $\overline{\Sigma}$ (recall the definition of $\overline{\Sigma}$ in Section 8.2.1). From M we reconstruct the initial 2-zone decomposition m by pasting vertices of degree 1 which are incident to the same face, and taking the dual map. From M we define a new rooted map on $\overline{\Sigma}$ in the following way: we start deleting recursively vertices of degree 1 which are not roots. Then we continue dissolving vertices of degree 2. The resulting map has $\beta(\Sigma)$ faces and all vertices have degree at least 3 (apart from root vertices, which have degree 1). The resulting map is called the *scheme associated* to M; we denote it by S_M. See Figure 8.4 for an example.

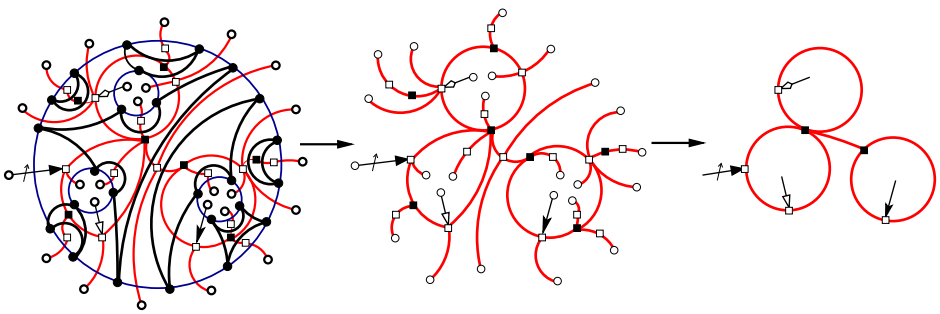

Figure 8.4: The construction of the scheme of an element in $\mathcal{P}_\Sigma$. We consider the dual of an irreducible 2-zone decomposition (leftmost figure). After deleting vertices of degree 1 recursively and dissolving vertices of degree 2, we obtain the associated scheme (rightmost figure).

An inverse construction can be done using maps over $\overline{\Sigma}$ and families of plane trees. Using these basic pieces, we can reconstruct all irreducible 2-zone decompositions. The details of this construction can be found in Section 8.8. Exploiting this decomposition and using singularity analysis (see Section 8.2.4 for the basic definitions), we get the following theorem (Γ denotes the classical Gamma function [117]):

Theorem 8.2 *Let Σ be a surface with boundary. Then the number $|\Pi_\Sigma(k)|$ verifies*

$$|\Pi_\Sigma(k)| \leq_{k\to\infty} \frac{C(\Sigma)}{\Gamma\left(3/2\gamma(\Sigma)+\beta(\Sigma)-3\right)} \cdot k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4} \cdot 4^k, \tag{8.2}$$

where $C(\Sigma)$ is a function depending only on Σ that is bounded by $\gamma(\Sigma)^{O(\gamma(\Sigma))}$.

The steps towards the proof of Theorem 8.2 are included in Sections 8.9, 8.10, and 8.11. Basically, we start characterizing the combinatorial decomposition in terms of plane trees. This combinatorial decomposition is exploited in Proposition 8.3 of Section 8.9 in order to count irreducible 2-zone decompositions. The constant $C(\Sigma)$ is related to the enumeration of cubic maps [42, 133]. Bounds for $C(\Sigma)$ are given in Section 8.10 (see Proposition 8.5). Finally, we prove in Section 8.11 that the asymptotic of $|\mathcal{R}_\Sigma(k)|$ coincides with the one obtained for irreducible 2-zone decompositions. The argument uses a double induction on the number of boundaries and the genus of the surface, and Lemma 8.3 of Section 8.5.

8.7.3 Additional constructions

In the previous section, we enumerated families of non-crossing partitions with boundary. In this section we first deal with a set of additional vertices that play the role of *apices* (cf. the last paragraph of Section 8.6). Secondly, we show how to extend the enumeration from non-crossing partitions to non-crossing packings. In both cases, we show that the modification over generating functions (GFs for short) does not depend on the surface Σ where non-crossing partitions are considered. The analysis consists in symbolic manipulation of GFs and application of singularity analysis over the resulting expressions.

The first problem can be stated in general as follows: for a sequence of positive numbers $\{p_{k,r}\}$, such that we know the GF $\sum_{k,r>0} p_{k,r} z^k u^r$, we want to estimate the value of $\sum_{r=1}^{k} r^l p_{k,r}$ for a fixed value l. This problem arises from the fact that we have a set of vertices (the *apices*), such that every vertex of the set can be associated to an arbitrary block of a non-crossing partition. The details of the analysis of this problem are done in Section 8.12. Basically, this problem only introduces a variation in the subexponential term of the asymptotic stated in Theorem 8.2. The second problem consists in generalizing from non-crossing partitions to non-crossing packings. In other words, for a fixed number of k vertices, fix an arbitrary subset of $i \leq k$ vertices, and consider the set of non-crossing partitions on Σ on this set of i vertices. This value is precisely $|\Pi_\Sigma(i)|$. Among the total set of k vertices, this set of i vertices can be chosen in $\binom{k}{i}$ ways. Hence, we want to estimate the sum $\sum_{i=0}^{k} \binom{k}{i} |\Pi_\Sigma(i)|$. Observe that this construction is quite close to Bell numbers, which count the number of ways a set of k elements can be partitioned into nonempty subsets. The details of the analysis can be found in Section 8.13. In this case, a combinatorial trick (Lemma 8.12) shows that the modification only affects the base of the exponential term.

Combining the univariate asymptotic obtained in Theorem 8.2 with the constructions described above (the details can be found in Propositions 8.7 and 8.8 in Sections 8.12 and 8.13, respectively) we obtain the following theorem, which gives the bound on the size of the tables when using surface cut decompositions:

Theorem 8.3 *Let $\overline{\Pi_{\Sigma,l}}(k)$ be the set of non-crossing partitions of Σ with k vertices and a set of l apices. Then the value $\sum_{i=0}^{k} \binom{k}{i} \left|\overline{\Pi_{\Sigma,l}}(k)\right|$ is upper-bounded, for large k, by*

$$\frac{C(\Sigma)}{2^{2+l}\Gamma\left(3/2\gamma(\Sigma)+\beta(\Sigma)-3\right)} \cdot k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4+l} \cdot 5^{k+1}, \tag{8.3}$$

where $C(\Sigma)$ is a function depending only on Σ that is bounded by $\gamma(\Sigma)^{O(\gamma(\Sigma))}$.

8.8 Enumeration of Non-crossing Partitions of the Disk and Related Constructions

In this section we introduce some terminology related to trees that arise as dual graphs of non-crossing partitions on the disk. Then, we use these concepts to obtain the number of non-crossing partitions of the disk with n vertices. At last, we introduce some families of related trees, which are used in the construction of the dual map of a non-crossing partition in a surface of higher genus.

The dual graph of a non-crossing partition is a tree, which is called the (non-crossing partition) tree associated to the non-crossing partition. Vertices of degree 1 (that is, the leafs of the tree) are called *danglings*. Vertices of the tree are called *block vertices* if they are associated to a block of the non-crossing partition. The remaining vertices are either *non-polygon vertices* or *danglings*. By construction, all vertices adjacent to a polygon vertex are non-polygon vertices. Conversely, each vertex adjacent to a non-polygon vertex is either a block vertex or a dangling. Graphically, we use the symbols ■ for block vertices, □ for non-polygon vertices and ∘ for danglings.

Denote by $\mathcal{T}$ the set of non-crossing partitions trees, and let $\mathbf{T} = \mathbf{T}(z,u) = \sum_{k,n>0} \mathbf{t}_{k,m} z^k u^m$ be the corresponding GF. The variable z marks danglings and u marks block vertices. We use also an auxiliary family $\mathcal{B}$, defined as the set of trees which are rooted at a non-polygon vertex. Let $\mathbf{B} = \mathbf{B}(z,u) = \sum_{k,n>0} \mathbf{b}_{k,m} z^k u^m$ be the associated GF. The next lemma gives the exact enumeration of $\mathcal{T}$ and $\mathcal{B}$.

Lemma 8.11 *The number of non-crossing trees counted by the number of danglings and block vertices is enumerated by*

$$\mathbf{T}(z,u) = \frac{1 - z(1-u) - \sqrt{(z(1-u)-1)^2 - 4zu}}{2zu}. \tag{8.4}$$

Furthermore, $\mathbf{B}(z,u) = z\mathbf{T}(z,u)$.

Proof: We establish combinatorial relations between $\mathcal{B}$ and $\mathcal{T}$, from which we deduce the desired result. First, observe that there is no restriction on the size of the blocks. Hence the degree of every block vertex is arbitrary. This condition is translated symbolically via the following relation: $\mathcal{T} = \blacksquare \times \mathrm{Seq}(\mathcal{B})$. Similarly, $\mathcal{B}$ can be written in the form $\mathcal{B} = \{\circ\} \times \mathrm{Seq}(\mathcal{T} \times \{\circ\})$.

These combinatorial conditions translate using Table 8.1 into the system of equations

$$\mathbf{T} = \frac{u}{1-\mathbf{B}}, \ \mathbf{B} = \frac{z}{1-z\mathbf{T}}.$$

If we substitute the expression of $\mathbf{B}$ in the first equation, one obtains that $\mathbf{T}$ satisfies the equation $z\mathbf{T}^2 + (z(1-u)-1)\mathbf{T} + u = 0$. The valid solution of this equation is (8.4). Solving the previous system of equations in terms of $\mathbf{B}$, we obtain that $\mathbf{B} = z\mathbf{T}$, as claimed. □

Observe that writing $u = 1$, we obtain that $\mathbf{T}(z) = \mathbf{T}(z,1) = \frac{1-\sqrt{1-4z}}{2z}$, and $\mathbf{B}(z) = \mathbf{B}(z,1) = z\mathbf{T}(z)$, and we recover the GF for Catalan numbers.

We need also a set of families of trees that are quite related to the previous ones. We call them *double trees*. A double tree is obtained in the following way: consider a path where we concatenate vertices of type ■ with vertices of type □. A double tree is a tree obtained by pasting on every internal vertex of type ■ a pair of elements of $\mathcal{T}$ (one at each side of the path), and similarly for internal vertices of type □. We say that a double tree is of type either ■ − ■ ■ − □, or □ − □ depending on the type of the ends of the path. An example for a double tree of type ■ − ■ is shown in Figure 8.5.

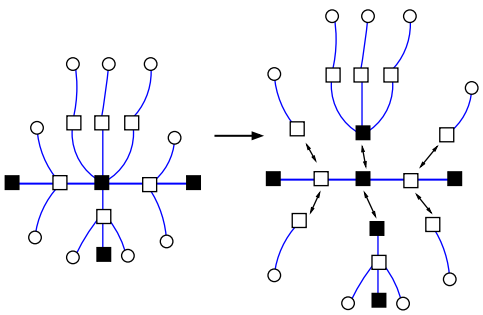

Figure 8.5: A double tree and its decomposition.

We denote these families by $\mathcal{T}_{■-■}$, $\mathcal{T}_{□-■}$, and $\mathcal{T}_{□-□}$, and the corresponding GF by $\mathbf{T}_1(z,u) = \mathbf{T}_1$, $\mathbf{T}_2(z,u) = \mathbf{T}_2$ and $\mathbf{T}_3(z,u) = \mathbf{T}_3$, respectively. Recall that in all cases z marks danglings and u marks block vertices. A direct application of the symbolic method provides a way to obtain explicit expressions for the previous GFs. The decomposition and the GFs of the three families is summarized in Table 8.2.

Family	Developement	Compact expression
$\mathcal{T}_{□-■}$	$1 + \frac{1}{u}\mathbf{B}^2\mathbf{T}^2 + \frac{1}{u^2}\mathbf{B}^4\mathbf{T}^4 + \dots$	$1/(1 - \mathbf{T}^2\mathbf{B}^2/u)$
$\mathcal{T}_{■-■}$	$\mathbf{B}^2 + \frac{1}{u}\mathbf{B}^4\mathbf{T}^2 + \frac{1}{u^2}\mathbf{B}^6\mathbf{T}^4 + \dots$	$\mathbf{B}^2/(1 - \mathbf{T}^2\mathbf{B}^2/u)$
$\mathcal{T}_{□-□}$	$\frac{1}{u}\mathbf{T}^2 + \frac{1}{u^2}\mathbf{B}^2\mathbf{T}^4 + \frac{1}{u^3}\mathbf{B}^4\mathbf{T}^6 + \dots$	$\frac{1}{u}\mathbf{T}^2/(1 - \mathbf{T}^2\mathbf{B}^2/u)$

Table 8.2: GFs for double trees.

To conclude, the family of *pointed* non-crossing trees $\mathcal{T}^\bullet$ is built pointing a dangling over each tree. In this case, the associated GF is $\mathbf{T}^\bullet = z\frac{\partial}{\partial z}\mathbf{T}$. Similar definitions can be done for the family $\mathcal{B}$. Pointing a dangling define a unique path between this distinguished dangling and the root of the tree.

8.9 Combinatorial Decomposition and Enumeration

Throughout this section, we use the notation and definitions introduced in Section 8.8 (i.e., families of trees, double trees and pointed trees, and the corresponding GFs). To simplify the notation, we denote by $\mathcal{P}_\Sigma(k,m)$ the set of irreducible 2-zone decompositions of Σ with

k vertices and m blocks. We write $p^{\Sigma}_{k,m}$ for the cardinal of this set. Let $p^{\Sigma}_{k} = \sum_{m>0} p^{\Sigma}_{k,m}$. The GF associated to the numbers $p^{\Sigma}_{k,m}$ is denoted by $\mathbf{P}_{\Sigma}(z,u)$. We denote by $\mathfrak{S}_{\Sigma}$ the set of rooted maps on $\overline{\Sigma}$ with $\beta(\Sigma)$ faces, whose vertices are bicolored (either ■ or □) and have degree at least 3. In particular, endpoints of a given edge can have the same color. This notation is used in Sections 8.11, 8.12, and 8.13. Observe that in our framework, each map has $\beta(\Sigma)$ roots, in contrast to the classical theory of enumeration of rooted maps (where a unique root is considered).

Applying Euler's formula for maps on $\overline{\Sigma}$ imples that $|\mathfrak{S}_{\Sigma}|$ is finite, because the number of faces is fixed. It is also obvious that if S_M is the scheme associated to a map M, then $S_M \in \mathfrak{S}_{\Sigma}$. These observations provide a way to establish a combinatorial bijection, that can be exploited to obtain the enumeration of $\mathcal{P}_{\Sigma}$. More concretely, each element M can be constructed from an element S of $\mathfrak{S}_{\Sigma}$ in the following way:

1. For an edge of S with both end-vertices of type ■, we paste a double tree of type ■ − ■ along it. Similar operations can be realized for edges with end-vertices $\{□, ■\}$ and $\{□, □\}$.
2. For a block vertex v of S, not incident with any root, we paste $\mathrm{d}(v)$ elements of $\mathcal{T}$ (identifying the roots of trees in $\mathcal{T}$ with v), one in each region determined by consecutive half-edges.
3. For a set of roots with an end-point in the same block vertex v, we paste an element of $\mathcal{T}^{\bullet}$ along each one of the roots (the marked leaf determines which is the dangling root). Over v we paste trees of $\mathcal{T}$ as we have done in the previous case. We do not paste trees between a root and a half-edge of S. A similar operation is done if the vertex is of type □.

This construction is shown for a concrete example in Figure 8.6.

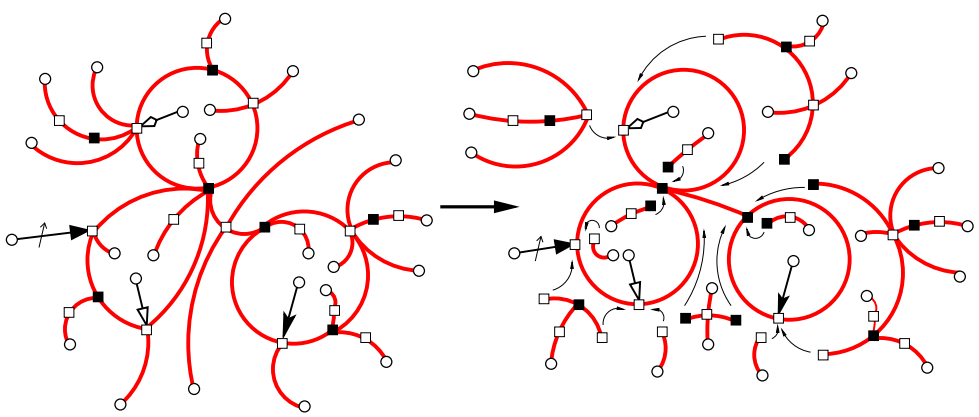

Figure 8.6: The decomposition into bicolored trees and the associated scheme.

Let us introduce some notation. Consider an element S of $\mathfrak{S}_{\Sigma}$. Let $v_1(S), v_2(S)$ be the set of block vertices and non-polygon vertices of S, respectively. Write $B(S), W(S)$ for the number of roots which are incident with a vertex of type ■ and □, respectively. In particular, $B(S) + W(S) = \beta(\Sigma)$. Denote by $e_1(S)$ the number of edges in S of type ■ − ■.

We similarly define $e_2(S)$ and $e_3(S)$ for edges of type $\square - \blacksquare$ and $\square - \square$, respectively. Observe that $e_1(S) + e_2(S) + e_3(S) + B(S) + W(S)$ is the number of edges of S, that is $e(S) = |E(S)|$. For a vertex x of S, denote by $r(x)$ the number of roots which are incident with it.

The previous decomposition provides a direct way to obtain the desired enumeration.

Proposition 8.3 *Let Σ be a surface with boundary. Then the coefficient $[z^k]\mathrm{P}_\Sigma(z,1)$ has an asymptotic expansion of the form*

$$[z^k]\mathrm{P}_\Sigma(z,1) = p_k^\Sigma =_{k\to\infty} \frac{C(\Sigma)}{\Gamma(-3\chi(\Sigma)/2+\beta(\Sigma))} k^{-3\chi(\Sigma)/2+\beta(\Sigma)-1} 4^k (1+O(k^{-1/2})), \tag{8.5}$$

where $C(\Sigma)$ is a function depending only on Σ.

Proof: According to the previous observations, $\mathrm{P}_\Sigma(z,u)$ can be written in the following form: for each $S \in \mathfrak{S}_\Sigma$, we replace edges (not roots) with double trees, roots with pointed trees, and vertices with sets of trees. More concretely, the GF we obtain is

$$\sum_{S\in\mathfrak{S}_\Sigma} u^{|v_1(S)|}\mathbf{T}_1^{e_1(S)}\mathbf{T}_2^{e_2(S)}\mathbf{T}_3^{e_3(S)}\left(\frac{\mathbf{T}}{u}\right)^{\sum_{x\in v_1(S)}(\mathrm{d}(x)-2r(x))}\mathbf{B}^{\sum_{y\in v_2(S)}(d(y)-2r(y))}\left(\frac{\mathbf{T}^\bullet}{u}\right)^{B(S)}(\mathbf{B}^\bullet)^{W(S)}. \tag{8.6}$$

Observe that terms $\mathbf{T}$ and $\mathbf{T}^\bullet$ appear divided by u. The reason is that we paste non-crossing trees through the root, which is a block vertex. In order to do it, we delete the corresponding block vertex, we paste the trees identifying their roots without counting the root, and finally we add the total number of block vertices (thus the term $u^{|v_1(S)|}$). To obtain the asymptotic behavior in terms of the number of danglings, we write $u = 1$ in Equation (8.6). To study the resulting GF, we need the expression of each factor of Equation (8.6) when we write $u = 1$. In Table 8.3 all the expressions are shown. This table is built from the expressions for $\mathbf{T}$ and $\mathbf{B}$ deduced in Lemma 8.11 and the expressions for double trees in Table 8.2. The GF in Equation (8.6) is a finite sum (a total of $|\mathfrak{S}_\Sigma|$

GF	Expression
$\mathbf{T}_1(z)$	$1/16(1-4z)^{-1/2} - 1/8(1-4z)^{1/2} + 1/16(1-4z)^{3/2}$
$\mathbf{T}_2(z)$	$1/4(1-4z)^{-1/2} + 1/2 + (1-4z)^{1/2}$
$\mathbf{T}_3(z)$	$z^2\left(1/16(1-4z)^{-1/2} - 1/8(1-4z)^{1/2} + 1/16(1-4z)^{3/2}\right)$
$\mathbf{T}(z)$	$1/(2z)(1-(1-4z)^{1/2})$
$\mathbf{B}(z)$	$1/2\left(1-(1-4z)^{1/2}\right)$
$\mathbf{T}^\bullet(z)$	$1/z(1-4z)^{-1/2} - 1/(2z^2)(1-(1-4z)^{-1/2})$
$\mathbf{B}^\bullet(z)$	$(1-4z)^{-1/2}$

Table 8.3: Univariate GF for all families of trees.

terms), so its singularity is located at $z = 1/4$ (since each addend has a singularity at this point). For each choice of S,

$$\mathbf{T}(z,1)^{\sum_{x\in v_1(S)}(\mathrm{d}(x)-2r(x))}\mathbf{B}(z,1)^{\sum_{y\in v_2(S)}(\mathrm{d}(y)-2r(y))} = \sum_{n=0}^{f(S)} f_n(z)(1-4z)^{n/2}, \tag{8.7}$$

where the positive integer $f(S)$ depends only on S, $f_n(z)$ are functions analytic at $z = 1/4$, and $f_0(z) \neq 0$ at $z = 1/4$. For the other multiplicative terms, we obtain

$$\mathbf{T}_1(z,1)^{e_1(S)}\mathbf{T}_2(z,1)^{e_2(S)}\mathbf{T}_3(z,1)^{e_3(S)}\mathbf{T}^\bullet(z,1)^{B(S)}\mathbf{B}^\bullet(z,1)^{W(S)} = G_S(z)(1-4z)^{-\frac{e(S)}{2}} + \ldots, \quad (8.8)$$

where $G_S(z)$ is an analytic function at $z = 1/4$. The reason of this fact is that each GF in the previous formula can be written in the form $p(z)(1-4z)^{-1/2} + \ldots$, where $p(z)$ is a function analytic at $z = 1/4$, and $e_1(S) + e_2(S) + e_3(S) + B(S) + W(S)$ is the total number of edges. Multiplying Equation (8.7) and Expression (8.8) we recover the contribution of a map S in $\mathrm{P}_\Sigma(z,1)$. More concretely, the contribution of a single map S to Equation (8.6) can be written in the form

$$g_S(z)(1-4z)^{-e(S)/2} + \ldots,$$

where $g_S(z)$ is an analytic function at $z = 1/4$. From Equation (8.1), the maps giving the greatest contribution to the asymptotic of p_k^Σ are the ones which maximize the value of $e(S)$. Applying Euler's formula (recall that all maps in $\mathfrak{S}_\Sigma$ have $\beta(\Sigma)$ faces) on $\overline{\Sigma}$ gives that these maps are the ones where each vertex have degree 3 (i.e., cubic maps). In particular, cubic maps with $\beta(\Sigma)$ faces and $\beta(\Sigma)$ roots have $2\beta(\Sigma) - 3\chi(\Sigma)$ edges. Hence, as a consequence of the Transfer Theorem for singularity analysis, the singular expansion of $\mathrm{P}_\Sigma(z,1)$ at $z = 1/4$ is

$$\mathrm{P}_\Sigma(z,1) =_{z\to 1/4} C(\Sigma)(1-4z)^{3\chi(\Sigma)/2-\beta(\Sigma)}\left(1 + O((1-4z)^{1/2})\right), \quad (8.9)$$

where $C(\Sigma) = \sum_{S\in\mathfrak{S}_\Sigma} g_S(1/4)$. Applying the Transfer Theorem in this expression yields the claimed result. □

8.10 Bounding $C(\Sigma)$ in Terms of Cubic Maps

In this section we obtain bounds for $C(\Sigma)$. We use the same notation as in Section 8.9. A more refined analysis over functions $g_S(z)$ provides upper bounds for $C(\Sigma)$. This is done in the following proposition:

Proposition 8.4 *The function $C(\Sigma)$ defined in Proposition 8.3 satisfies*

$$C(\Sigma) \leq 2^{\beta(\Sigma)}|\mathfrak{S}_\Sigma|. \quad (8.10)$$

Proof: For each $S \in \mathfrak{S}_\Sigma$, we obtain bounds for $g_S(1/4)$. We use Table 8.4, which is a simplification of Table 8.3. We are only concerned now about the constant term of each GF. Table 8.4 brings the following information: the greatest contribution from double trees, trees, and families of pointed trees comes from $\mathcal{T}_{\square-\blacksquare}$, $\mathcal{T}$, and $\mathcal{T}^\bullet$, respectively. The constants are 1/4, 2, and 4, respectively. Each cubic map has $2\beta(\Sigma) - 3\chi(\Sigma)$ edges ($\beta(\Sigma)$ of them being roots) and $\beta(\Sigma) - 2\chi(\Sigma)$ vertices ($\beta(\Sigma)$ of them being incident with roots). This characterization provides the following upper bound for $g_S(1/4)$:

$$g_S(1/4) \leq \left(\frac{1}{4}\right)^{2\beta(\Sigma)-3\chi(\Sigma)-\beta(\Sigma)} 2^{-3\cdot 2\chi(\Sigma)+\beta(\Sigma)} 4^{\beta(\Sigma)} = 2^{\beta(\Sigma)}. \quad (8.11)$$

□

GF	Expression	Developement at $z = 1/4$
$\mathbf{T}_1(z)$	$1/16(1-4z)^{-1/2} + \dots$	$1/16(1-4z)^{-1/2} + \dots$
$\mathbf{T}_2(z)$	$1/4(1-4z)^{-1/2} + \dots$	$1/4(1-4z)^{-1/2} + \dots$
$\mathbf{T}_3(z)$	$z^2/16(1-4z)^{-1/2} + \dots$	$1/256(1-4z)^{-1/2} + \dots$
$\mathbf{T}(z)$	$1/(2z) + \dots$	$2 + \dots$
$\mathbf{B}(z)$	$1/2 + \dots$	$1/2 + \dots$
$\mathbf{T}^\bullet(z)$	$1/z(1-4z)^{-1/2} + \dots$	$4(1-4z)^{-1/2} + \dots$
$\mathbf{B}^\bullet(z)$	$(1-4z)^{-1/2}$	$(1-4z)^{-1/2}$

Table 8.4: A simplification of Table 8.3 used in Proposition 8.3.

The value of $\mathfrak{S}_\Sigma$ can be bounded using the results in [42, 133]. Indeed, Gao shows in [133] that the number of rooted cubic maps with n vertices in an orientable surface of genus[6] g is asymptotically equal to

$$t_g \cdot n^{5(g-1)/2} \cdot (12\sqrt{3})^n,$$

where the constant t_g tends to 0 as g tends to ∞ [42]. A similar result is also stated in [133] for non-orientable surfaces. By duality, the number of rooted cubic maps in a surface $\overline{\Sigma}$ of genus $\chi(\Sigma)$ with $\beta(\Sigma)$ faces is asymptotically equal to $t_{\chi(\Sigma)} \cdot \beta(\Sigma)^{5(\chi(\Sigma)-1)/2} \cdot (12\sqrt{3})^{\beta(\Sigma)}$. This value is clearly bounded by $\gamma(\Sigma)^{O(\gamma(\Sigma))}$.

To conclude, we observe that the elements of $\mathfrak{S}_\Sigma$ are obtained from rooted cubic maps with $\beta(\Sigma)$ faces by adding a root on each face different from the root face. Observe that each edge is incident with at most two faces, and that the total number of edges is $-3\chi(\Sigma)$. Consequently, the number of ways of rooting a cubic map with $\beta(\Sigma) - 1$ unrooted faces is bounded by $\binom{-6\chi(\Sigma)}{\beta(\Sigma)-1}$, which is bounded by $\gamma(\Sigma)^{O(\gamma(\Sigma))}$.

By the above discussion, the following proposition holds.

Proposition 8.5 *The constant $C(\Sigma)$ verifies*

$$C(\Sigma) \leq t_{\chi(\Sigma)} \cdot \beta(\Sigma)^{5(\chi(\Sigma)-1)/2} \cdot (12\sqrt{3})^{\beta(\Sigma)} \binom{-6\chi(\Sigma)}{\beta(\Sigma)-1} 2^{\beta(\Sigma)}.$$

In particular, $C(\Sigma) = \gamma(\Sigma)^{O(\gamma(\Sigma))}$.

Combining Propositions 8.3 and 8.5, we obtain that

$$p_k^\Sigma \leq_{k\to\infty} \frac{C(\Sigma)}{\Gamma(3/2\gamma(\Sigma) + \beta(\Sigma) - 3)} \cdot k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4} \cdot 4^k, \tag{8.12}$$

where $C(\Sigma) = \gamma(\Sigma)^{O(\gamma(\Sigma))}$ is a function depending only on Σ.

[6] the *genus* $g(\Sigma)$ of an orientable surface Σ is defined as $g(\Sigma) = \gamma(\Sigma)/2$ (see [171]).

8.11 Reducibility vs Irreducibility

In this section we use the same notation as in Section 8.8. For conciseness we use the notation $a(\Sigma)$ to denote the constant term which appears in all the asymptotic expressions in Section 8.8.

For a non-irreducible regular element s of $\mathcal{R}_\Sigma$ (recall Lemma 8.10) there is a non-contractible cycle $\mathbb{S}^1$ contained in a white 2-dimensional region of s. Additionally, s induces a regular 2-zone decomposition over the surface $\Sigma \nearrow \mathbb{S}^1 = \Sigma'$, which can be irreducible or not. By Lemma 8.9, each element of $\mathcal{R}_{\Sigma'}$ defines an element on $\mathcal{R}_\Sigma$. To prove that irreducible 2-zone decompositions over Σ give the maximal contribution to the asymptotic, we apply a double induction argument on the pair $(\gamma(\Sigma), \beta(\Sigma))$. The critical point is the initial step, which corresponds to $\gamma(\Sigma) = 0$:

Proposition 8.6 *Let Σ be a surface obtained from the sphere deleting β disjoints disks. Then*

$$|\mathcal{R}_\Sigma(k)| \leq_{k\to\infty} |\mathcal{P}_\Sigma(k)|.$$

Proof: Induction on β. The case $\beta = 1$ corresponds to the disk. We deduced in Section 8.8 the exact expression for $\mathrm{P}_\Sigma(z,u)$ (see Equation (8.4)). In this case the equality $|\mathcal{R}_\Sigma(k)| = |\mathcal{P}_\Sigma(k)|$ holds for every value of k. Let us consider now the case $\beta = 2$, which corresponds to the cylinder. From Equation (8.12), the number of irreducible 2-zone decompositions over the cylinder verifies

$$|\mathcal{P}_\Sigma(k)| =_{k\to\infty} a(\Sigma)\cdot k \cdot 4^k(1 + O(k^{-1/2})). \tag{8.13}$$

Let us calculate upper bounds for the asymptotic of non-irreducible 2-zone decompositions on a cylinder. A non-contractible cycle $\mathbb{S}^1$ on a cylinder separates it into a pair of cylinders. In other words $\Sigma' = \Sigma \nearrow \mathbb{S}^1$ is a pair of disks. The asymptotic in this case is of the form $[z^k]\mathbf{T}(z,1)^2 =_{k\to\infty} O(k^{-3/2}4^k)$. The subexponential term in Equation (8.13) is greater, so the claim of the proposition holds for $\beta = 1$.

Let us proceed to apply the inductive step. Let $\beta > 1$ be the number of boundaries of Σ. A non-contractible cycle $\mathbb{S}^1$ always separates Σ into two connected components, namely Σ_1 and Σ_2. Let $\beta_1, \beta_2 < \beta$ be the number of boundaries of Σ_1 and Σ_2, respectively. By induction hypothesis,

$$|\mathcal{R}_{\Sigma_j}(k)| \leq_{k\to\infty} |\mathcal{P}_{\Sigma_j}(k)|,$$

for $j = 1, 2$. Consequently, we only need to deal with irreducible decompositions of Σ_1 and Σ_2. The GF of 2-zone regular decompositions that reduces to decompositions over Σ_1 and Σ_2 has the same asymptotic as $\mathrm{P}_{\Sigma_1}(z,1)\cdot \mathrm{P}_{\Sigma_2}(z,1)$. The estimate of its coefficients is

$$[z^k]\mathrm{P}_{\Sigma_1}(z,1)\cdot \mathrm{P}_{\Sigma_2}(z,1) \leq a(\Sigma_1)\cdot a(\Sigma_2)[z^k](1-4z)^{-5/2\beta_1+3}\cdot(1-4z)^{-5/2\beta_2+3} =_{k\to\infty} O\left(k^{5/2\beta-7}\cdot 4^k\right).$$

Consequently, the above term is smaller than p_k^Σ when k is large enough (the value is of the form $\left(k^{5/2\beta-4}4^k\right)$, and does not depend on how Σ is cut. □

The next step is to adapt the previous argument to surfaces of genus greater than 0. Let Σ be a surface with boundary and Euler genus $\gamma(\Sigma)$. Consider a non-contractible cycle $\mathbb{S}^1$ and the resulting surface $\Upsilon = \Sigma \nearrow \mathbb{S}^1$. Two situations can occur:

1. Υ is connected and $\beta(\Upsilon) = \beta(\Sigma)$. In this case, the Euler genus has been decreased by either 1 if the cycle is one-sided or by 2 if the cycle is two-sided. This result appears as Lemma 4.2.4 in [171].
2. The resulting surface is not connected, $\Upsilon = \Upsilon_1 \sqcup \Upsilon_2$. In this case, the total number of boundaries is $\beta(\Upsilon) = \beta(\Upsilon_1) + \beta(\Upsilon_2)$. By Lemma 8.3, $\gamma(\Sigma) = \gamma(\Upsilon_1) + \gamma(\Upsilon_2) - 2$.

The induction argument distinguishes between these two cases: if $\Upsilon = \Sigma \rightarrowtail \mathbb{S}^1$ is connected, by induction on the genus, $|\mathcal{R}_\Upsilon(k)| \leq_{k\to\infty} |\mathcal{P}_\Upsilon(k)|$. Additionally, by Expression (8.12), an upper bound for $|\mathcal{P}_\Upsilon(k)|$ is

$$[z^k]\mathrm{P}_\Upsilon(z,1) = a(\Upsilon)\cdot k^{3/2\gamma(\Upsilon)+\beta(\Upsilon)-4}\cdot 4^k(1+O(k^{-1/2})) =_{k\to\infty} O\left(k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4}\cdot 4^k\right).$$

If Υ is not connected, then $\Upsilon = \Upsilon_1 \sqcup \Upsilon_2$, $\beta(\Sigma) = \beta(\Upsilon) - 2 = \beta(\Upsilon_1) + \beta(\Upsilon_2)$, and $\gamma(\Sigma) = \gamma(\Upsilon_1) + \gamma(\Upsilon_2)$. Again, by induction hypothesis we only need to look at the irreducible ones. Consequently,

$$[z^k]\mathrm{P}_{\Upsilon_1}(z,1)\mathrm{P}_{\Upsilon_2}(z,1) = a(\Upsilon_1)\cdot a(\Upsilon_2)[z^k](1-4z)^{-3/2(\gamma(\Upsilon_1)+\gamma(\Upsilon_2))-(\beta(\Upsilon_1)+\beta(\Upsilon_2))+6}.$$

The exponent of $(1-4z)$ can be written as $-3/2\gamma(\Sigma) - \beta(\Sigma) + 6$. Consequently, the value $[z^k]\mathrm{P}_{\Upsilon_1}(z,1)\mathrm{P}_{\Upsilon_2}(z,1)$ is bounded, for k large enough, by

$$k^{3/2\gamma(\Sigma)+\beta(\Sigma)-6-1}\cdot 4^k = k^{3/2\gamma(\Sigma)+\beta(\Sigma)-7}\cdot 4^k = O\left(k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4}\cdot 4^k\right).$$

Hence the contribution is smaller than the one given by $|\mathcal{P}_\Sigma(k)|$, as claimed.

8.12 Dealing with a Set of Apices

Due to the definition of surface cut decomposition, we need to modify the family $\mathcal{P}_\Sigma$ of irreducible 2-zone decompositions in the following way: consider a set of l vertices $\{\overline{1}, \overline{2}, \ldots, \overline{l}\} = [\bar{l}]$ disjoint from the set of vertices over Σ. This set of vertices is called set of *apices*. For every value of k we want to count the number of pairs (s_r, f), where $s_r \in \mathcal{P}_\Sigma(k)$ has r blocks, and f is an arbitrary application $f : [\bar{l}] \to [r]$. The number of such pairs is $\sum_{r=1}^{k} r^l p_{k,r}^\Sigma$ (recall that the number of irreducible 2-zone decompositions with k vertices and r blocks is $p_{k,r}^\Sigma$, and the associated GF is $\mathrm{P}_\Sigma(z,u)$). The aim of this section is to obtain estimates for this sum. This problem can be stated in the following equivalent way: let $\mathbf{F}(z,u)$ be a GF with expansion $\mathbf{F}(z,u) = \sum_{k,r\geq 0} f_{k,r} z^k u^r$, such that $[z^k u^r]\mathbf{F}(z,u) = 0$ if $r > k$. For a non-negative integer l, we want to estimate the sum

$$\sum_{r=0}^{k} r^l f_{k,r} = [z^k] \sum_{k\geq 0} \sum_{r=0}^{\infty} r^l f_{k,r} z^k u^r \Big|_{u=1},$$

for k large enough. Let Θ be the pointing operator on the second variable: $\Theta\mathbf{F}(z,u) = u\frac{\partial}{\partial u}\mathbf{F}(z,u) = u\mathbf{F}_u(z,u)$. Applying l times the operator Θ over $\mathbf{F}(z,u)$ gives $\Theta^l\mathbf{F}(z,u) = \sum_{k,m\geq 0}^{\infty} m^l f_{k,r} z^k u^m$, so our problem consists in estimating $[z^k]\Theta^l\mathbf{F}(z,u)\big|_{u=1}$. The strategy we use to obtain the estimate consists in simplifying this expression up to a function from which we know to get the asymptotic. Firstly, observe that Θ^l can be written as $\sum_{i=1}^{l} q_i(u)\frac{\partial^i}{\partial u^i}$, where $q_i(u)$ is a polynomial on u. For $i = l$, the value of $q_l(u)$ is u^l. We

show that the greatest contribution to the enumeration comes from the term $i = l$. As a consequence, we only need to deal with $u^l \frac{\partial^l}{\partial u^l}\mathbf{F}(z,u)$. To do this, it is also convenient to observe the following: for $b < a$ positive real numbers, the asymptotic of $(1-4z)^{-a}$ is greater than the one for $(1-4z)^{-b}$: from the Transfer Theorem for singularity analysis (Equation (8.1)), both GFs have an exponential growth of the form 4^k. However, their asymptotic growth is not the same: while the first one has a subexponential growth of the form k^{a-1}, the second one is of the form k^{b-1}, which is smaller. Generalizing this to a linear combination of terms of the form $(1-4z)^{-a_i}$, where a_i is a positive real number, the asymptotic of the whole function comes from the value a_i with the greatest modulus.

Let us return to study $\mathrm{P}_\Sigma(u,z)$. Observe that GFs for double trees can be factorized in the following way (consult Table 8.2):

$$\frac{\mathbf{G}(z,u)}{1-\frac{1}{u}\mathbf{T}^2\mathbf{B}^2/u} = \frac{-2uz^2\mathbf{G}(z,u)}{((u+1)\,z-1)\,\sqrt{(z(1-u)-1)^2-4zu}+z^2(u-1)^2-2z(u+1)+1},$$

where $\mathbf{G}(z,u)$ is either 1, $\mathbf{T}^2/u$ or $\mathbf{B}^2$. For conciseness, write $\mathbf{f} = (z(1-u)-1)^2-4zu$, $\mathbf{g} = ((u+1)\,z-1)$ and $\mathbf{h} = z^2(u-1)^2-2z(u+1)+1$. The previous formula can be written in the form $-2uz^2\mathbf{G}(z,u)/(\mathbf{h}+\mathbf{g}\sqrt{\mathbf{f}})$. Additionally, $\mathbf{g}(z,1) = 2z-1$, $\mathbf{f}(z,1) = 1-4z$ and $\mathbf{h}(z,1) = 1-4z$. For $u = 1$, the smallest singularity of the function is located at $z = 1/4$, where function $\sqrt{\mathbf{f}}$ ceases to be analytic. Consequently, the source of the singularity on a double tree comes exclusively from the term $\sqrt{\mathbf{f}}$. Furthermore, when we write $u = 1$, the smallest singularity of every derivative (with respect to u) of $-2uz^2\mathbf{G}(z,u)/(\mathbf{h}+\mathbf{g}\sqrt{\mathbf{f}})$ is located at $z = 1/4$, because the denominator is always the same (possibly with a greater exponent). A similar argument applies to the families of pointed trees (same behavior, $\mathbf{f}^{-1/2}$).

Taking into account this, and rationalizing the previous expressions, Expression (8.6) can be written in the form

$$\mathrm{P}_\Sigma(z,u) = \sum_{S\in\mathfrak{S}_\Sigma}\left(\mathbf{g}_S(z,u)\mathbf{f}^{-e(S)/2}+\dots\right),$$

where "…" means that the exponent of the other terms is smaller in modulus (and they give smaller contributions to the asymptotic enumeration). Observe that $\mathbf{g}(z,u)$ is analytic at $(z,u) = (1/4,1)$, and satisfies that $\mathbf{g}_S(z,1) = g_S(z)$. This presentation for $\mathrm{P}_\Sigma(z,u)$ is the correct one to deal with the operator Θ^l: observe that the greatest contribution (using the Transfer Theorem) comes from cubic maps, which are the ones with maximize the number of edges (i.e., the value $e(S) = 3\gamma(\Sigma)+2\beta(\Sigma)-6$). For conciseness on the formulas, until the end of this section we write $e = 3\gamma(\Sigma)+2\beta(\Sigma)-6$.

We need to study the derivative $\frac{\partial^l}{\partial u^l}\left(\mathbf{g}_S(z,u)\mathbf{f}^{-e/2}\right)$, which is the main contribution of each cubic map. When we apply this derivative over $\mathbf{g}_S(z,u)\mathbf{f}^{-e/2}$, the greatest contribution comes from $\mathbf{g}_S(z,u)u^l\frac{\partial}{\partial u^l}\mathbf{f}^{-e/2}$, because this term maximizes the exponent (in modulus) of the singular term. In this case, the singular term with greatest exponent corresponds to

$$u^l\mathbf{g}_S(z,u)\frac{(-1)^l\Gamma(e/2+l)}{\Gamma(e/2)}\frac{(\mathbf{f}_u(z,u))^l}{\mathbf{f}(z,u)^{e/2+l}},$$

where $\mathbf{f}_u(z,u)$ is the derivative of $\mathbf{f}$ with respect to u. Writing $u=1$, the previous expression is simplified into

$$g_S(z)\frac{\Gamma(e/2+l)}{\Gamma(e/2)}\frac{(2z)^l}{(1-4z)^{e/2+l}}.$$

To estimate the value of the k-th coefficient of the previous GF, we apply the Transfer Theorem for singularity analysis (Equation (8.1)), obtaining

$$g_S(1/4)\frac{1}{\Gamma(e/2)}2^{-l}\cdot k^{e/2+l-1}\cdot 4^k(1+O(k^{-1/2})).$$

To conclude, recall that this above term is the contribution of a single cubic map. Summing over all cubic maps, we obtain the following proposition:

Proposition 8.7 *Let $p^\Sigma_{k,r}$ be the number of irreducible 2-zone decompositions of Σ. For a fixed positive integer l, the following asymptotic approximation holds:*

$$\sum_{r=1}^{k} r^l p^\Sigma_{k,r} =_{k\to\infty} \frac{C(\Sigma)\cdot 2^{-l}}{\Gamma(3\gamma(\Sigma)/2+\beta(\Sigma)-3)}\cdot k^{3\gamma(\Sigma)/2+\beta(\Sigma)-4+l}\cdot 4^k(1+O(k^{-1/2})), \tag{8.14}$$

where an upper bound for $C(\Sigma)$ is stated in Proposition 8.4.

8.13 Bell Structures: from Partitions to Packings

For a fixed number of k vertices, fix an arbitrary subset of $i\le k$ vertices, and consider the set of non-crossing partitions over Σ using this set of i vertices. This value is precisely p^Σ_i. This set of i vertices can be chosen in $\binom{k}{i}$ ways. Consequently, we want to estimate the sum $\sum_{i=0}^k \binom{k}{i}p^\Sigma_i$. Observe that this construction is quite close to Bell numbers, which count the number of ways a set of k elements can be partitioned into nonempty subsets. The main result of this section uses the following combinatorial trick:

Lemma 8.12 *Let $\boldsymbol{A}(z)=\sum_{n>0}a_nz^n$. Then the sum $\sum_{i=0}^n\binom{n}{i}a_i$ is $[z^n]\frac{1}{1-z}\boldsymbol{A}(\frac{z}{1-z})$.*

Proof: It is a consequence of the Taylor development of $\frac{1}{1-z}\mathbf{A}(\frac{z}{1-z})$ and the relation $\frac{z^n}{(1-z)^{n+1}}=\sum_{i=0}^\infty\binom{n+i}{i}z^{n+i}$, which can be proved by induction. □

Consequently, $\mathrm{P}_\Sigma(z,1)$ is modified via Lemma 8.12 to obtain the GF for the numbers $\sum_{i=0}^k\binom{k}{i}p^\Sigma_i$. The singularity of $\mathrm{P}_\Sigma(z,1)$ is located at $z=1/4$, and therefore the singularity of $\frac{1}{1-z}\mathrm{P}_\Sigma(z/(1-z),1)$ is located at $z=1/5$. Its singular behavior (i.e., the singular exponent) is the same as the one for $\mathrm{P}_\Sigma(z,1)$. The modification is made only on the position of the singularity, and not on its nature.

Summarizing, the estimate of $\sum_{i=0}^k\binom{k}{i}p^\Sigma_i$ for k big enough has exponential term equal to 5^k, and subexponential term equal to the one of p^Σ_k. In other words, we have proved the following proposition:

Proposition 8.8 *The following estimate holds:*

$$\frac{\sum_{i=0}^k\binom{k}{i}p^\Sigma_i}{p^\Sigma_k} =_{k\to\to\infty}\left(\frac{5}{4}\right)^{k+1}\left(1+O(k^{-1/2})\right).$$

8.14 Conclusions

Our results can be summarized as follows.

Theorem 8.4 *Given a problem P belonging to Category (C) in a graph G embedded in a surface of Euler genus γ, with $\mathbf{bw}(G) \leq k$, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by $2^{O(k)} \cdot k^{O(\gamma)} \cdot \gamma^{O(\gamma)}$.*

As we mentioned, the problems tackled in [99] are those in Category (B), which are included in Category (C). As a result of this, we reproduce all the results of [99]. Moreover, as our approach does not use planarization, our analysis provides algorithms where the dependence on the Euler genus γ is better than the one in [99]. In particular, the running time of the algorithms in [99] is $2^{O(\gamma \cdot \mathbf{bw} + \gamma^2 \cdot \log(\mathbf{bw}))} \cdot n$, while in our case the running time is $2^{O(\mathbf{bw} + \gamma \cdot \log(\mathbf{bw}) + \gamma \cdot \log \gamma)} \cdot n$.

Dynamic programming is important for the design of subexponential exact or parameterized algorithms. Using the fact that bounded-genus graphs have branchwidth at most $O(\sqrt{\gamma \cdot n})$ [124], we derive the existence of exact algorithms in $O^*(2^{O(\sqrt{\gamma n} + \gamma \cdot \log(\gamma \cdot n))})$ steps for all problems in Category (C). Moreover, using bidimensionality theory (see [94, 96]), on can derive $2^{O(\gamma \cdot \sqrt{k} + \gamma \cdot \log(\gamma \cdot k))} \cdot n^{O(1)}$ step parameterized algorithms for all bidimensional problems in Category (C).

A natural extension of our results is to consider more general classes of graphs than bounded-genus graphs. This has been done in [101] for problems in Category (B), where the tables of the algorithms encode pairings of the middle set. To extend these results for problems in Category (C) (where tables encode subsets of the middle set), using the planarization approach of [101], appears to be a quite complicated task. We believe that our surface-oriented approach could be more successful in this direction.

Notice that Categories (A), (B), and (C) can be seen as the first levels of a more general hierarchy of dynamic programming algorithms designed for gradually more complicated combinatorial problems. For instance, higher level classes of algorithms can be defined for tables encoding connected pairings (or even connected packings) of subsets of the middle set. In a sense, what we prove in this chapter is the collapse of the time bounds in Category (C) to those in Category (A) when inputs are topologically restricted. It seems to be an interesting task to define such a hierarchy and to check whether this collapse extends to its higher levels.

Part IV

Conclusions and Further Research

The conclusions and avenues for further research corresponding to each chapter of this thesis have been given in page 52 (Chapter 1), page 71 (Chapter 2), page 97 (Chapter 3), page 119 (Chapter 4), page 151 (Chapter 5), page 170 (Chapter 6), page 181 (Chapter 7), and page 216 (Chapter 8). From a more global point of view, here we conclude the thesis and briefly propose possible lines for further research in Sections IV.1 and IV.2, respectively.

IV.1 Final conclusions

This thesis consisted of two main parts: traffic grooming and degree-constrained subgraph problems. Originally motivated by an optimization problem in optical networks (Chapters 1-4), we got interested in progressively more general problems. Namely, in Chapters 5-7 we focused on degree-constrained subgraph problems, and in Chapter 8 we provided a framework to deal with problems whose solutions can be codified by subsets of vertices. Let us now give some more details about the problems we considered.

Traffic grooming. In Chapter 1 we studied the traffic grooming problem on rings and paths with a general traffic pattern. We proved hardness results and provided approximation algorithms. In Chapter 2 we modeled the traffic grooming in unidirectional rings in the case when the request graph has bounded maximum degree and the objective is to design a network being able to support any request graph satisfying the degree constraints. We were able to settle the (asymptotically) optimal number of ADMs at each node for almost all values of the grooming factor and the maximum degree. In Chapters 3 and 4 we focused on the ring topology with an all-to-all traffic pattern. Namely, in Chapter 3 we studied the bidirectional ring, providing optimal solutions for infinite families of values of the size of the network and the grooming factor. In Chapter 4 we dealt with the unidirectional ring and two grooming factors C and C' that alternate dynamically. Using tools from combinatorial designs, we found the optimal switching cost for $C = 4$ and $C' \in \{1, 2, 3\}$, as well as the optimal switching cost under the constraint of using the minimum number of wavelengths.

In view of our results, we observe that there is a trade-off between the *generality* of the considered problem (like the request graph, the topology, or the grooming factor) and the *quality* of the solutions that one can expect to obtain in a reasonable computation time. Namely, the more general the setting is, the further we are from an optimal solution. This phenomenon is not surprising, as traffic grooming is an NP-hard problem and, unless P = NP, optimal solutions of an NP-hard problem cannot be found in polynomial time.

Degree-constrained subgraph problems. In the second part of this thesis we applied a variety of approaches to study degree-constrained subgraph problems. In Chapter 5 we provided hardness results and approximation algorithms for several such problems, using for instance the error amplification technique and randomized algorithms. We studied in Chapter 6 the parameterized complexity of these problems in order the better understand their apparent hardness. We proved $W[1]$-hardness results using parameterized reductions

and provided FPT algorithms using refined dynamic programming and structural results from graph minors theory. In Chapters 7 and 8 we focused on the case when the input is topologically restricted. Namely, in Chapter 7 we obtained subexponential parameterized and exact algorithms for a family of degree-constrained subgraph problems on planar graphs, using bidimensionality theory combined with novel dynamic programming techniques. Finally, we provided in Chapter 8 a framework for the design of algorithms with single-exponential dependence on branchwidth for a broad class of problems on graphs embedded in surfaces. This framework introduces a new type of branch decomposition, called *surface cut decomposition*, and uses tools from topological graph theory together with the symbolic method and singularity analysis from analytic combinatorics.

IV.2 Further Research

Concerning traffic grooming, we have mentioned along the thesis that the unidirectional ring with all-to-all traffic in an important special case. So far, the formulas of the minimum number of ADMs as a function of the network size have been found for values of the grooming factor up to seven (see page 36). Finding the optimal cost for each value of the grooming factor involves complicated tailor-made constructions. Although finding these formulas may yield new combinatorial designs and interesting insights, it does not make sense to aim at solving all the (infinite) cases of the grooming factor one by one. It would be more relevant to conceive a machinery (probably, with the help of a computer) able to generate the cost formulas in reasonable time for each fixed value of the grooming factor.

In this thesis we moved towards more and more general problems. How far can one go in this direction? The answer probably lies on the so-called *algorithmic meta-theorems*. One of the most notorious such theorems, the fundamental theorem of Courcelle [86], states that graph properties definable in monadic second-order logic can be decided in linear time on graphs of bounded treewidth (or equivalently, bounded branchwidth, see page 22). This was the first in a series of algorithmic meta-theorems. More recent examples of such meta-theorems state that all first-order definable properties of planar graphs can be decided in linear time [128] and that all first-order definable optimization problems on classes of graphs with excluded minors can be approximated in polynomial time to any given approximation ratio [90]. The term "meta-theorem" refers to the fact that these results do not describe algorithms for specific problems, but for whole families of problems, whose definition typically has a logical and a structural (usually graph-theoretical) component. For example, Courcelle's theorem [86] is about monadic second-order logic on graphs of bounded treewidth. The general form of algorithmic meta-theorems is: "All problems definable in a *certain logic* on a certain class of *structures* can be solved *efficiently*". We refer the reader to the excellent survey of Grohe [142].

Many of the meta-theorems are tightly linked with graph minor theory. Recently, results from graph minor theory have been combined with algorithmic techniques that had originally been developed for planar graphs to obtain polynomial time approximation schemes and FPT algorithms for many standard optimization problems on families of graphs with excluded minors. The fascinating topic of algorithmic meta-theorems is receiving increasing attention during the last years and seems to be a promising research field.

Appendix A

Permutation Routing and (ℓ, k)-routing on Plane Grids

The packet routing problem plays an essential role in communication networks and has been extensively studied during the last decades. It involves how to transfer data from some origins to some destinations within a reasonable amount of time. In the (ℓ, k)-routing problem, each node can send at most ℓ packets and receive at most k packets. Permutation routing is the particular case $\ell = k = 1$. In the r-central routing problem, all nodes at distance at most r from a fixed node v want to send a packet to v. The goal is to minimize the number of time steps required to route all packets to their respective destinations, under the constraint that each link can be crossed simultaneously by no more than one packet.

Keywords: Packet routing, distributed algorithm, permutation routing, (ℓ, k)-routing, plane grids, communication networks, shortest path, oblivious algorithm.

A.1 Permutation Routing on Triangular Grids

To measure the routing capability of an interconnection network, the *permutation routing* problem is usually used as the metric. This problem has been studied in a wide diversity of scenarios, such as mobile ad hoc networks, cube-connected cycle networks, wireless and radio networks, all-optical networks, reconfigurable meshes, or circulant graphs.

In [J5, C21] we study this problem in hexagonal networks, i.e., finite subgraphs of a triangular grid, which is a widely used network in practical applications. We present the first optimal permutation routing algorithm on full-duplex hexagonal networks, using the addressing scheme described by Nocetti, Stojmenović, and Zhang in [175]. Our algorithm is fully distributed and surprisingly simple. Furthermore, we prove that the algorithm is oblivious and translation invariant.

A.2 (ℓ, k)-routing on Plane Grids

In [J3, B7] we study the permutation routing, the r-central routing, and the general (ℓ, k)-routing problems on plane grids (that is, square, triangular, and hexagonal grids). We use the *store-and-forward* Δ-port model, and we consider both full- and half-duplex networks. We first survey the existing results in the literature about packet routing, with special emphasis on (ℓ, k)-routing on plane grids. Our main contributions are the following:

1. Tight permutation routing algorithms on full-duplex hexagonal grids, and half duplex triangular and hexagonal grids.

2. Tight r-central routing algorithms on triangular and hexagonal grids.

3. Tight (k, k)-routing algorithms on square, triangular and hexagonal grids.

4. Good approximation algorithms (in terms of the running time) for (ℓ, k)-routing on square, triangular and hexagonal grids, together with new lower bounds on the running time of any algorithm using shortest path routing.

All these algorithms are completely distributed, i.e., can be implemented independently at each node. Finally, we also formulate the (ℓ, k)-routing problem as a Weighted Edge Coloring problem on bipartite graphs.

Appendix B

Label Space Minimization in GMPLS Networks

All-Optical Label Switching (AOLS) is an approach to transparently route packets all-optically, allowing a speed-up of the forwarding. This very promising technology for the future Internet applications also brings new constraints and, consequently, new problems have to be addressed. Indeed, as the forwarding functions are implemented directly at the optical domain, a specific correlator is needed for each optical label processed in the node. Therefore, it is of major importance to reduce the number of employed correlators in every node, hence reducing the number of *labels* (as referred in the literature). The most promising scheme to manage the control plane of these optical networks is Generic MultiProtocol Label Switching (GMPLS). Therefore, for reducing the total number of labels in routers, solutions deployed by GMPLS for reducing the number of labels, such as label merging or label stacking, have to be studied.

Keywords: GMPLS, label stacking, dynamic programming, approximation algorithms, hardness of approximation, hypergraph.

B.1 GMPLS Label Stacking on the Path

In [C11] we study the problem of routing a set of requests in AOLS networks with the aim of minimizing the number of labels required to ensure the forwarding. In order to spare the label space, we consider label stacking, allowing the configuration of tunnels. We study particularly this network design problem when the network is a line. We provide an exact algorithm (based on a dynamic programming approach) for the case in which all the requests have a common source and present some approximation algorithms and heuristics when an arbitrary number of sources are distributed over the line. We analyze by simulations the performance of our proposed algorithms and compare them with previous ones.

B.2 Designing Hypergraphs Layouts to GMPLS Routing Strategies

In [C12] we continue the work initiated in [C11] and study the problem of routing a set of requests in AOLS networks using GMPLS technology, with the aim of minimizing the number of labels required to ensure the forwarding. We first formalize the problem by associating to each routing strategy a logical hypergraph whose hyperarcs are dipaths of the physical graph, called *tunnels* in GMPLS terminology. Such a hypergraph is called a *hypergraph layout*, to which we assign a cost function given by its physical length plus the total number of hops traveled by the traffic. Minimizing the cost of the design of an AOLS network can then be expressed as finding a minimum cost hypergraph layout.

We prove hardness results for the problem, namely for general directed networks we prove that it is NP-hard to find a $C \log n$-approximation, where C is a a positive constant and n is the number of nodes of the network. For symmetric directed networks, we prove that the problem is APX-hard. These hardness results hold even is the traffic instance is a partial broadcast. On the other hand, we provide an $O(\log n)$-approximation algorithm to the problem for a general symmetric network. Finally, we focus on the case where the physical network is a path, providing a polynomial-time dynamic programming algorithm for a bounded number of sources, thus extending the algorithm given in [C11] for a single source.

Appendix C

Tolerance Graphs

A graph $G = (V, E)$ on n vertices is a *tolerance graph* if there is a set $I = \{I_i \mid i = 1, \ldots, n\}$ of closed intervals on the real line and a set $T = \{t_i \mid i = 1, \ldots, n\}$ of positive real numbers, called *tolerances*, such that for any two vertices $v_i, v_j \in V$, $v_i v_j \in E$ if and only if $|I_i \cap I_j| \geq \min\{t_i, t_j\}$. These sets of intervals and tolerances form a *tolerance representation* of G. If G has a tolerance representation such that $t_i \leq |I_i|$ for $i = 1, \ldots, n$, then G is called a *bounded tolerance graph* and its representation is a *bounded tolerance representation.*

Tolerance graphs were introduced by Golumbic and Monma in [139], mainly motivated by the need to solve scheduling problems in which resources that would be normally used exclusively, like rooms or vehicles, can tolerate some sharing among users. Since then, tolerance graphs have been widely studied in the literature, as they naturally generalize both interval graphs (when all tolerances are equal) and permutation graphs (when $|I_i| = t_i$ for $i = 1, \ldots, n$), see [139]. For more details, see the book of Golumbic and Trenk [140].

Keywords: Tolerance graphs, parallelogram graphs, intersection model, minimum coloring, maximum clique, weighted independent set.

C.1 A New Intersection Model and Improved Algorithms

In [J4,C16] we propose the first non-trivial intersection model for general tolerance graphs, given by three-dimensional parallelepipeds, which extends the widely known intersection model of parallelograms in the plane that characterizes the class of bounded tolerance graphs. Apart from being important on its own, this new representation also enables us to improve the time complexity of three problems on tolerance graphs. Namely, we present optimal $O(n \log n)$ algorithms for computing a minimum coloring and a maximum clique, and an $O(n^2)$ algorithm for computing a maximum weight independent set in a tolerance graph with n vertices, thus improving the best known running times $O(n^2)$ and $O(n^3)$ for these problems, respectively.

C.2 The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete

Several efficient algorithms for optimization problems that are NP-hard in general graphs have been designed for tolerance graphs. In spite of this, the complexity of the recognition of tolerance graphs – namely, the problem of deciding whether a given graph is a tolerance graph – as well as the complexity of the recognition of their main subclass of bounded tolerance graphs, have been the most fundamental open problems on this class of graphs (cf. the book on tolerance graphs [140]) since their introduction in 1982 [139]. Therefore, all existing algorithms assume that, along with the input tolerance graph, a tolerance representation of it is given. The only result about the complexity of recognizing tolerance and bounded tolerance graphs is that they have a (non-trivial) polynomial sized tolerance representation, hence the problems of recognizing tolerance and bounded tolerance graphs are in the class NP [147].

If we replace in the definition of tolerance graphs the operator *min* by the operator *max*, we obtain the class of *max-tolerance* graphs, which also finds natural applications in several contexts. The recognition of max-tolerance graphs is known to be NP-hard [157]. Unfortunately, the structure of max-tolerance graphs differs significantly from that of tolerance graphs (max-tolerance graphs are not even perfect as it is the case of tolerance graphs, since they can contain induced C_5's [157]), so the technique used in [157] does not carry over to tolerance graphs.

In [C17][1] we prove that both recognition problems are NP-complete, even in the case where the input graph is a trapezoid graph. The presented results are surprising because, on the one hand, most subclasses of perfect graphs admit polynomial recognition algorithms and, on the other hand, bounded tolerance graphs were believed to be efficiently recognizable as they are a natural special case of trapezoid graphs, which can be recognized in polynomial time [69]. For our reduction we extend the notion of an acyclic orientation of permutation and trapezoid graphs. Our main tool is a new algorithm that transforms a given trapezoid graph into a permutation graph, while preserving this new acyclic orientation property.

[1] Full version available at `http://sunsite.informatik.rwth-aachen.de/Publications/AIB/2009/2009-06.pdf`.

Appendix D

Miscellaneous

D.1 Edge-simple Circuits Through 10 Ordered Vertices in Square Grids

A *circuit* in a simple undirected graph $G = (V, E)$ is a sequence of vertices $\{v_1, v_2, \ldots, v_{k+1}\}$ such that $v_1 = v_{k+1}$ and $\{v_i, v_{i+i}\} \in E$ for $i = 1, \ldots, k$. A circuit C is said to be *edge-simple* if no edge of G is used twice in C. An edge-simple circuit is also called *closed trail* in the literature. The existence of a circuit through a prescribed set of vertices or edges has been an important graph-theoretical question for many years.

In [C14] we study the following problem: which is the largest integer k such that, given any subset of k ordered vertices of an infinite square grid, there exists an edge-simple circuit visiting the k vertices in the prescribed order? We prove that $k = 10$. To this end, we first provide a counterexample implying that $k < 11$. To show that $k \geq 10$, we introduce a methodology, based on the notion of core graph, to reduce drastically the number of possible vertex configurations, and then we test each one of the resulting configurations with an ILP solver.

Keywords: Square grid, edge-simple circuit, prescribed vertices, ILP solver.

D.2 Self-duality of Branchwidth in Graphs of Bounded Genus

A *surface* is a connected compact 2-manifold without boundaries. A surface Σ can be obtained, up to homeomorphism, by adding $\mathbf{eg}(\Sigma)$ *crosscaps* to the sphere. $\mathbf{eg}(\Sigma)$ is called the *Euler genus* of Σ.

A graph parameter is *self-dual* in some class of graphs embeddable in some surface if its value does not change in the dual graph more than a constant factor. Self-duality has been examined for several width-parameters, such as branchwidth, pathwidth, and treewidth.

In [C19] we give a direct proof of the self-duality of branchwidth (denoted **bw**) in graphs embedded in some surface. In this direction, we prove that $\mathbf{bw}(G^*) \leq 6 \cdot \mathbf{bw}(G) + 2g - 4$ for any graph G embedded in a surface of Euler genus g.

Keywords: Graphs on surfaces, branchwidth, duality, polyhedral embedding.

D.3 7-[3]coloring Algorithm for Triangle-free Hexagonal Graphs

Given a graph G and a demand function $p : V(G) \to \mathbb{N}$, a proper n-$[p]$*coloring* is a mapping $f : V(G) \to \{1, \ldots, n\}$ such that $|f(v)| \geq p(v)$ for any vertex $v \in V(G)$ and $f(v) \cap f(u) = \emptyset$ for any pair of adjacent vertices u and v. The least integer n for which a proper n-$[p]$coloring exists, $\chi(G)$, is called the *multichromatic number* of G. Finding the multichromatic number of induced subgraphs of the triangular lattice (called *hexagonal graphs*) has important applications in cellular networks. The *weighted clique number* of a graph G, $\omega(G)$, is the maximum weight of a clique in G, where the *weight* of a clique is the total demand of its vertices. McDiarmid and Reed [169] conjectured that $\chi(G) \leq (9/8)\omega(G) + o(1)$ for triangle-free hexagonal graphs.

In [S29][1] we provide an algorithm to find a 7-[3]coloring of triangle-free hexagonal graphs, which implies that $\chi(G) \leq (7/6)\omega(G) + o(1)$. Our result constitutes a much shorter alternative to the inductive proof of Havet [146] and improves the short proof of Sudeep and Vishwanathan [197], who proved the existence of a 14-[6]coloring.

Keywords: Graph algorithm, approximation algorithm, graph coloring, frequency planning, cellular networks.

[1] Available until acceptance at `http://www-sop.inria.fr/members/Ignasi.Sauvalls/multicoloring.pdf`.

List of Figures

Index

Bibliography

Personal Bibliography

International Journals

[J1] O. Amini, S. Pérennes, and I. Sau. Hardness and Approximation of Traffic Grooming. *Theoretical Computer Science*, 410(38-40):3751–3760, 2009.

[J2] J.-C. Bermond, C. J. Colbourn, L. Gionfriddo, G. Quattrocchi, and I. Sau. Drop Cost and Wavelength Optimal Two-Period Grooming with Ratio 4. *SIAM Journal on Discrete Mathematics*, 2010. To appear.

[J3] F. Huc, I. Sau, and J. Žerovnik. (ℓ, k)-Routing on Plane Grids. *Journal of Interconnection Networks*, 10(1-2):27–57, 2009.

[J4] G. B. Mertzios, I. Sau, and S. Zaks. A New Intersection Model and Improved Algorithms for Tolerance Graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1800–1813, 2009.

[J5] I. Sau and J. Žerovnik. An Optimal Permutation Routing Algorithm on Full-Duplex Hexagonal Networks. *Discrete Mathematics and Theoretical Computer Science*, 10(3):49–62, 2008.

Book Chapters

[B6] T. Cinkler, D. Coudert, M. Flammini, G. Monaco, L. Moscardelli, X. Muñoz, I. Sau, M. Shalom, and S. Zaks. *Studies in Broadband, Optical, Wireless, and Ad Hoc Networks*, chapter Traffic Grooming: Combinatorial Results and Practical Resolutions. EATCS Texts in Theoretical Computer Science. Springer, 2009.

[B7] I. Sau and J. Žerovnik. *Studies in Broadband, Optical, Wireless, and Ad Hoc Networks*, chapter Permutation Routing and (ℓ, k)-Routing on Plane Grids. EATCS Texts in Theoretical Computer Science. Springer, 2009.

International Conferences

[C8] O. Amini, D. Peleg, S. Pérennes, I. Sau, and S. Saurabh. Degree-Constrained Subgraph Problems: Hardness and Approximation Results. In *Proceedings of Workshop on Approximation and On-line Algorithms (ALGO/WAOA)*, pages 29–42, volume 5426 of LNCS, 2008.

[C9] O. Amini, S. Pérennes, and I. Sau. Hardness and Approximation of Traffic Grooming. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC)*, pages 561–573, volume 4835 of LNCS, 2007.

[C10] O. Amini, I. Sau, and S. Saurabh. Parameterized Complexity of the Smallest Degree-Constrained Subgraph Problem. In *Proceedings of International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 13–29, volume 5008 of LNCS, 2008.

[C11] J.-C. Bermond, D. Coudert, J. Moulierac, S. Pérennes, H. Rivano, I. Sau, and F. Solano Donado. MPLS label stacking on the line network. In *Proceedings of IFIP Networking*, volume 5550 of *LNCS*, pages 809–820, 2009.

[C12] J.-C. Bermond, D. Coudert, J. Moulierac, S. Perennes, I. Sau, and F. Solano Donado. Designing Hypergraph Layouts to GMPLS Routing Strategies. In *Proceedings of the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, LNCS, 2009.

[C13] J.-C. Bermond, D. Coudert, X. Muñoz, and I. Sau. Traffic Grooming in Bidirectional WDM Ring Networks. In *Proceedings of IEEE-LEOS ICTON*, volume 3, pages 19–22, 2006.

[C14] D. Coudert, F. Giroire, and I. Sau. Edge-Simple Circuits Through 10 Ordered Vertices in Square Grids. In *Proceedings of the 20th International Workshop on Combinatorial Algorithms (IWOCA)*, pages 134–145, volume 5874 of LNCS, 2009.

[C15] Z. Li and I. Sau. Graph Partitioning and Traffic Grooming with Bounded Degree Request Graph. In *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, 2009. To appear. Best student paper award.

[C16] G. B. Mertzios, I. Sau, and S. Zaks. A New Intersection Model and Improved Algorithms for Tolerance Graphs. In *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, 2009. To appear.

[C17] G. B. Mertzios, I. Sau, and S. Zaks. The Recognition of Tolerance and Bounded Tolerance Graphs. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2010. To appear.

[C18] X. Muñoz and I. Sau. Traffic Grooming in Unidirectional WDM Rings with Bounded Degree Request Graph. In *Proceedings of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 300–311, volume 5344 of LNCS, 2008.

[C19] I. Sau and D. M. Thilikos. On Self-Duality of Branchwidth in Graphs of Bounded Genus. In *Proceedings of the 8th Cologne Twente Workshop on Graphs and Combinatorial Optimization (CTW)*, pages 19–22, 2009.

[C20] I. Sau and D. M. Thilikos. Subexponential Parameterized Algorithms for Bounded-Degree Connected Subgraph Problems on Planar Graphs. In *Proceedings of DIMAP workshop on Algorithmic Graph Theory (AGT)*, volume 32 of *Electronic Notes in Discrete Mathematics*, pages 59–66, 2009.

[C21] I. Sau and J. Žerovnik. Optimal permutation routing on mesh networks. In *Proceedings of International Network Optimization Conference (INOC)*, 2007. 6 pages.

National Conferences

[N22] O. Amini, S. Pérennes, and I. Sau. Hardness of approximating the traffic grooming problem. In *Proceedings of 9ème Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)*, pages 45–48, 2007.

Submitted for Publication

[S23] O. Amini, D. Peleg, S. Pérennes, I. Sau, and S. Saurabh. On the Approximability of some Degree-constrained Subgraph Problems. Manuscript submitted to journal, 2009.

[S24] O. Amini, I. Sau, and S. Saurabh. Parameterized Complexity of Finding Small Degree-constrained Subgraphs. Manuscript submitted to journal, 2009.

[S25] J.-C. Bermond, X. Muñoz, and I. Sau. Traffic Grooming in Bidirectional WDM Ring Networks. Manuscript submitted to journal, 2009.

[S26] X. Muñoz, Z. Li, and I. Sau. Edge-partitioning Regular Graphs for Ring Traffic Grooming with a Priori Placement of the ADMs. Manuscript submitted to journal, 2009.

[S27] J. Rué, I. Sau, and D. M. Thilikos. Dynamic Programming for Graphs on Surfaces. Manuscript submitted to conference, 2009.

[S28] I. Sau and D. M. Thilikos. Subexponential Parameterized Algorithms for Degree-Constrained Subgraph Problems on Planar Graphs. Manuscript submitted to journal, 2009.

[S29] I. Sau, P. Šparl, and J. Žerovnik. 7-[3]coloring Algorithm for Triangle-free Hexagonal Graphs. Manuscript submitted to journal, 2009.

General Bibliography

[30] L. Addario-Berry, K. Dalal, and B. Reed. Degree constrained subgraphs. *Discrete Applied Mathematics*, 156(7):1168–1174, 2008.

[31] P. Alimonti and V. Kann. Hardness of Approximating Problems on Cubic Graphs. In *Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC)*, volume 1203 of *LNCS*, pages 288–298, 1997.

[32] N. Alon, S. Friedland, and G. Kalai. Every 4-regular graph plus an edge contains a 3-regular subgraph. *Journal of Combinatorial Theory Series B*, 37:92–93, 1984.

[33] N. Alon, S. Friedland, and G. Kalai. Regular subgraphs of almost regular graphs. *Journal of Combinatorial Theory Series B*, 37:79–91, 1984.

[34] N. Alon, V. Teague, and N. C. Wormald. Linear Arboricity and Linear k-Arboricity of Regular Graphs. *Graphs and Combinatorics*, 17(1):11–16, 2001.

[35] N. Alon, R. Yuster, and U. Zwick. Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs. In *Proceedings of the 26th annual ACM symposium on Theory of Computing (STOC)*, pages 326–335, 1994.

[36] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 7–15, 2001.

[37] R. Andersen and K. Chellapilla. Finding Dense Subgraphs with Size Bounds. In *Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph (WAW)*, pages 25–37, volume 5427 of LNCS, 2009.

[38] R. P. Anstee. Minimum vertex weighted deficiency of (g, f)-factors: a greedy algorithm. *Discrete Applied Mathemathics*, 44(1-3):247–260, 1993.

[39] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey. *BIT*, 25(1):2–23, 1985.

[40] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25:272–289, 1996.

[41] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Perennes, and U. Vaccaro. Graph problems arising from Wavelength–Routing in All–Optical Networks. In *Proceedings of Workshop on Optics and Computer Science (WOCS)*, 1997.

[42] E. A. Bender, Z. Gao, and L. B. Richmond. The map asymptotics constant t_g. *Electronic Journal of Combinatorics*, 15(1):R51, 8 psges, 2008.

[43] C. Berge. *Graphs and Hypergraphs*. North-Holland, 1973.

[44] J.-C. Bermond, L. Braud, and D. Coudert. Traffic Grooming on the Path. *Theoretical Computer Science*, 384(2-3):139–151, 2007.

[45] J.-C. Bermond and S. Ceroi. Minimizing SONET ADMs in unidirectional WDM rings with grooming ratio 3. *Networks*, 41(2):83–86, 2003.

[46] J.-C. Bermond, C. J. Colbourn, D. Coudert, G. Ge, A. C. H. Ling, and X. Muñoz. Traffic grooming in unidirectional WDM rings with grooming ratio $C = 6$. *SIAM Journal on Discrete Mathematics*, 19(2):523–542, 2005.

[47] J.-C. Bermond, C. J. Colbourn, A. C. H. Ling, and M. L. Yu. Grooming in unidirectional rings: $K_4 - e$ designs. *Discrete Mathematics*, 284:67–72, 2004.

[48] J.-C. Bermond and D. Coudert. Traffic Grooming in Unidirectional WDM Ring Networks using Design Theory. In *Proceedings of IEEE International Conference on Communications (ICC)*, volume 2, pages 1402–1406, 2003.

[49] J.-C. Bermond and D. Coudert. *The CRC Handbook of Combinatorial Designs (2nd edition)*, volume 42 of *Discrete Mathematics and Its Applications*, chapter VI.27, Grooming, pages 493–496. CRC Press, C.J. Colbourn and J.H. Dinitz edition, 2006.

[50] J.-C. Bermond, D. Coudert, and B. Lévêque. Approximations for All-to-all Uniform Traffic Grooming on Unidirectional Rings. *Journal of Interconnection Networks*, 9(4):471–486, 2008.

[51] J.-C. Bermond, D. Coudert, and X. Muñoz. Traffic Grooming in Unidirectional WDM Ring Networks: the all-to-all unitary case. In *Proceedings of the 7th IFIP Working Conference on Optical Network Design and Modelling*, pages 1135–1153, 2003.

[52] J.-C. Bermond, D. Coudert, and M.-L. Yu. On DRC-Covering of K_n by cycles. *Journal of Combinatorial Designs*, 11(2):100–112, 2003.

[53] J.-C. Bermond, J.-L. Fouquet, M. Habib, and B. Péroche. On linear k-arboricity. *Discrete Mathematics*, 52(2-3):123–132, 1984.

[54] J.-C. Bermond and C. Peyrat. Induced Subgraphs of the Power of a Cycle. *SIAM Journal on Discrete Mathematics*, 2(4):452–455, 1989.

[55] O. Bernardi and J. Rué. Counting simplicial decompositions in surfaces with boundaries. Manuscript available at `http://lanl.arxiv.org/abs/0901.1608`.

[56] R. Berry and E. Modiano. Reducing electronic multiplexing costs in SONET/WDM rings with dynamically changing traffic. *IEEE Journal on Selected Areas in Communications*, 18:1961–1971, 2000.

[57] A. Björklund and T. Husfeldt. Finding a Path of Superlogarithmic Length. *SIAM Journal on Computing*, 32(6):1395–1402, 2003.

[58] H. L. Bodlaender. Dynamic Programming on Graphs with Bounded Treewidth. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 105–118, volume 317 of LNCS, 1988.

[59] H. L. Bodlaender. A Tourist Guide through Treewidth. *Acta Cybernetica*, 11(1-2):1–22, 1993.

[60] B. Bollobás and G. Brightwell. Long Cycles in Graphs with no Subgraphs of Minimal Degree 3. *Discrete Mathematics*, 75:47–53, 1989.

[61] V. Bonifaci, U. D. Iorio, and L. Laura. The complexity of uniform Nash equilibria and related regular subgraph problems. *Theoretical Computer Science*, 401(1-3):144–152, 2008.

[62] D. Bryant, P. Adams, and M. Buchanan. A survey on the existence of G-designs. *Journal of Combinatorial Designs*, 16:373–410, 2008.

[63] S. Cabello and B. Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discrete and Computational Geometry*, 37(2):213–235, 2007.

[64] A. Caprara and R. Rizzi. Packing triangles in bounded degree graphs. *Information Processing Letters*, 84(4):175–180, 2002.

[65] D. M. Cardoso, M. Kamiński, and V. Lozin. Maximum k-regular induced subgraphs. *Journal of Combinatorial Optimization*, 14(4):455–463, 2007.

[66] Y. Caro and J. Schönheim. Decompositions of trees into isomorphic subtrees. *Ars Combininatorica*, 9:119–130, 1980.

[67] P. A. Catlin. Supereulerian graphs: a survey. *Journal of Graph Theory*, 16(2):177–196, 1992.

[68] L. S. Chandran. A high girth graph construction. *SIAM Journal on Discrete Mathematics*, 16(3):366–370, 2003.

[69] F. Cheah and D. Corneil. On the structure of trapezoid graphs. *Discrete Applied Mathematics*, 66(2):109–133, 1996.

[70] F. Cheah and D. G. Corneil. The Complexity of Regular Subgraph Recognition. *Discrete Applied Mathematics*, 27:59–68, 1990.

[71] A. L. Chiu and E. H. Modiano. Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. *IEEE/OSA Journal of Lightwave Technology*, 18(1):2–12, 2000.

[72] B. Chor, M. Fellows, M. A. Ragan, I. Razgon, F. Rosamond, and S. Snir. Connected Coloring Completion for General Graphs: Algorithms and Complexity. In *Proceedings of the 13th Annual International Computing and Combinatorics Conference (COCOON)*, pages 75–85, volume 4598 of LNCS, 2007.

[73] T. Chow and P. Lin. Private communication.

[74] T. Chow and P. Lin. The ring grooming problem. *Networks*, 44(3):194–202, 2004.

[75] V. Chvátal, H. Fleischner, J. Sheehan, and C. Thomassen. Three-regular Subgraphs of Four Regular Graphs. *Journal of Graph Theory*, 3:371–386, 1979.

[76] T. Cinkler. Traffic- and λ-grooming. *IEEE Network*, 17(2):16–21, 2003.

[77] C. Colbourn and J. Dinitz, editors. *Handbook of Combinatorial Designs*. Number 0. Chapman & Hall/CRC, 2nd edition, 2006.

[78] C. J. Colbourn, H.-L. Fu, G. Ge, A. C. H. Ling, and H.-C. Lu. Minimizing SONET ADMs in Unidirectional WDM Rings with Grooming Ratio Seven. *SIAM Journal on Discrete Mathematics*, 23(1):109–122, 2008.

[79] C. J. Colbourn, A. C. H. Ling, and G. Quattrocchi. Minimum embedding of P_3-designs into $(K_4 - e)$-designs. *Journal of Combinatorial Designs*, 11:352–366, 2003.

[80] C. J. Colbourn, G. Quattrocchi, and V. R. Syrotiuk. Grooming for two-period optical networks. *Networks*, 52(4):307–324, 2008.

[81] C. J. Colbourn, G. Quattrocchi, and V. R. Syrotiuk. Lower bounds for two-period grooming via linear programming duality. *Networks*, 52(4):299–306, 2008.

[82] C. J. Colbourn and A. Rosa. Quadratic leaves of maximal partial triple systems. *Graphs and Combinatorics*, 2:317–337, 1986.

[83] C. J. Colbourn and A. Rosa. *Triple systems*. Oxford University Press, Oxford and New York, 1999.

[84] C. J. Colbourn and P.-J. Wan. Minimizing Drop Cost for SONET/WDM Networks with 1/8 Wavelength Requirements. *Networks*, 37(2):107–116, 2001.

[85] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley and Sons, New York, 1998.

[86] B. Courcelle. The Monadic Second-Order Logic of Graphs: Definable Sets of Finite Graphs. In *Proceedings of the 14th International Workshop on Graph-theoretic Concepts in Computer Science (WG)*, volume 344 of *LNCS*, pages 30–53, 1988.

[87] P. Crescenzi and V. Kann. A compendium of NP optimization problems. Accessible at `http://www.nada.kth.se/~viggo/wwwcompendium`.

[88] G. Călinescu, O. Frieder, and P.-J. Wan. Minimizing electronic line terminals for automatic ring protection in general WDM optical networks. *IEEE Journal of Selected Areas on Communications*, 20(1):183–189, 2002.

[89] A. Dawar, M. Grohe, and S. Kreutzer. Locally Excluding a Minor. In *Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 270–279, 2007.

[90] A. Dawar, M. Grohe, S. Kreutzer, and N. Schweikardt. Approximation schemes for first-order definable optimisation problems. In *Proceedings of the 21st IEEE Symposium on Logic in Computer Science (LICS)*, pages 411–420, 2006.

[91] D. de Werra. Equitable colorations of graphs. *RAIRO R-3*, pages 3–8, 1971.

[92] E. Demaine, M. Hajiaghayi, and K. C. Kawarabayashi. Algorithmic Graph Minor Theory: Decomposition, Approximation and Coloring. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 637–646, 2005.

[93] E. Demaine and M. T. Hajiaghayi. Equivalence of Local Treewidth and Linear Local Treewidth and its Algorithmic Applications. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 840–849, 2004.

[94] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.

[95] E. D. Demaine and M. Hajiaghayi. Bidimensionality: New Connections between FPT Algorithms and PTASs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 590–601, 2005.

[96] E. D. Demaine, M. Hajiaghayi, and D. M. Thilikos. The Bidimensional Theory of Bounded-Genus Graphs. *SIAM Journal on Discrete Mathematics*, 20(2):357–371, 2006.

[97] R. Diestel. *Graph Theory*. Springer-Verlag, 2005.

[98] F. Dorn. *Designing Subexponential Algorithms: Problems, Techniques and Structures*. PhD thesis, University of Bergen, 2007.

[99] F. Dorn, F. V. Fomin, and D. M. Thilikos. Fast Subexponential Algorithm for Non-local Problems on Graphs of Bounded Genus. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 4059 of *LNCS*, pages 172–183, 2006.

[100] F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 15–27, volume 4596 of LNCS, 2007.

[101] F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan structures and dynamic programming in H-minor-free graphs. In *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 631–640, 2008.

[102] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Branch Decompositions. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, pages 95–106, volume 3669 of LNCS, 2005.

[103] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York, 1999.

[104] R. Dutta, A. E. Kamal, and G. N. Rouskas, editors. *Traffic Grooming for Optical Networks: Foundations, Techniques and Frontiers.* Optical Networks. Springer, 2008.

[105] R. Dutta and N. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks*, 1(1):73–89, 2000.

[106] R. Dutta and N. Rouskas. Traffic Grooming in WDM Networks: Past and Future. *IEEE Network*, 16(6):46–56, 2002.

[107] T. Eilam, S. Moran, and S. Zaks. Lightpath arrangement in survivable rings to minimize the switching cost. *IEEE Journal of Selected Areas on Communications*, 20(1):172–182, 2002.

[108] D. Eppstein. Diameter and Tree-width in Minor-closed Graph Families. *Algorithmica*, 27(3-4):275–291, 200.

[109] D. Eppstein. Dynamic Generators of Topologically Embedded Graphs. In *Proceedings of the 14th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 599–608, 2003.

[110] L. Epstein and A. Levin. Better Bounds for Minimizing SONET ADMs. In *Proceedings of the Workshop on Approximation and On-line Algorithms (WAOA)*, pages 281–294, volume 3351 of LNCS, 2004.

[111] P. Erdős, R. J. Faudree, A. Gyárfás, and R. H. Schelp. Cycles in Graphs Without Proper Subgraphs of Minimum Degree 3. *Ars Combinatorica*, 25(B):195–201, 1988.

[112] P. Erdős, R. J. Faudree, C. C. Rousseau, and R. H. Schelp. Subgraphs of Minimal Degree k. *Discrete Mathematics*, 85(1):53–58, 1990.

[113] P. Erdös and H. Sachs. Reguläre graphe gegebener taillenweite mit minimaler knotenzahl. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 12:251–257, 1963.

[114] U. Feige, G. Kortsarz, and D. Peleg. The Dense k-Subgraph Problem. *Algorithmica*, 29(3):410–421, 2001.

[115] M. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the Parameterized Complexity of Multiple-Interval Graph Problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

[116] P. Flajolet and M. Noy. Analytic combinatorics of non-crossing configurations. *Discrete Mathematics*, 204(1-3):203–229, 1999.

[117] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.

[118] M. Flammini, G. Monaco, L. Moscardelli, M. Shalom, and S. Zaks. Approximating the traffic grooming problem in tree and star networks. *Journal of Parallel and Distributed Computing*, 68(7):939–948, 2008.

[119] M. Flammini, G. Monaco, L. Moscardelli, M. Shalom, and S. Zaks. Approximating the Traffic Grooming Problem with Respect to ADMs and OADMs. In *Proceedings of the 14th International Conference on Parallel and Distributed Computing (Euro-Par)*, pages 920–929, 2008.

[120] M. Flammini, L. Moscardelli, M. Shalom, and S. Zaks. Approximating the traffic grooming problem. *Journal of Discrete Algorithms*, 6(3):472–479, 2008.

[121] M. Flammini, M. Shalom, and S. Zaks. On minimizing the number of adms in a general topology optical network. In *Proceedings of the 20th International Workshop on Distributed Algorithms (DISC)*, 2006.

[122] M. Flammini, M. Shalom, and S. Zaks. On minimizing the number of adms - tight bounds for an algorithm without preprocessing. *Journal of Parallel and Distributed Computing*, 67(4):448–455, 2007.

[123] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer Verlag, 2006.

[124] F. V. Fomin and D. M. Thilikos. Fast Parameterized Algorithms for Graphs on Surfaces: Linear Kernel and Exponential Speed-Up . In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *LNCS*, pages 581–592, 2004.

[125] F. V. Fomin and D. M. Thilikos. New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory*, 51(1):53–81, 2005.

[126] F. V. Fomin and D. M. Thilikos. Dominating Sets in Planar Graphs: Branch-Width and Exponential Speed-Up. *SIAM Journal on Computing*, 36(2):281–309, 2006.

[127] F. V. Fomin and D. M. Thilikos. On Self Duality of Pathwidth in Polyhedral Graph Embeddings. *Journal of Graph Theory*, 55(42-54), 2007.

[128] M. Frick and M. Grohe. Deciding First-Order Properties of Locally Tree-Decomposable Structures. *Journal of the ACM*, 48(6):1184–1206, 2001.

[129] H. L. Fu and C. A. Rodger. Forest leaves and 4-cycles. *Journal of Graph Theory*, 33:161–166, 2000.

[130] M. Fürer and B. Raghavachari. Approximating the minimum-degree spanning tree to within one from the optimal degree. In *Proceedings of the 3rd annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 317–324, 1992.

[131] M. Fürer and B. Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994.

[132] H. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th annual ACM symposium on Theory of Computing (STOC)*, pages 448–456, 1983.

[133] Z. Gao. The number of rooted triangular maps on a surface. *Journal of Combinatorial Theory, Series B*, 52(2):236–249, 1991.

[134] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman, 1979.

[135] O. Gerstel, P. Lin, and G. Sasaki. Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings. In *IEEE INFOCOM*, pages 94–101, 1998.

[136] O. Gerstel, R. Ramaswami, and G. Sasaki. Cost Effective Traffic Grooming in WDM Rings. In *Proceedings of IEEE INFOCOM*, pages 69–77, 1998.

[137] M. X. Goemans. Minimum Bounded-Degree Spanning Trees. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 273–282, 2006.

[138] O. Goldschmidt, D. Hochbaum, A. Levin, and E. Olinick. The SONET edge-partition problem. *Networks*, 41(1):13–23, 2003.

[139] M. Golumbic and C. Monma. A generalization of interval graphs with tolerances. In *Proceedings of the 13th Southeastern Conference on Combinatorics, Graph Theory and Computing (SCCGTC), Congressus Numerantium 35*, pages 321–331, 1982.

[140] M. Golumbic and A. Trenk. *Tolerance Graphs.* Cambridge Studies in Advanced Mathematics, 2004.

[141] M. Grohe. Local Tree-width, Excluded Minors and Approximation Algorithms. *Combinatorica*, 23(4):613–632, 2003.

[142] M. Grohe. Logic, graphs, and algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(091), 2007.

[143] Q.-P. Gu and H. Tamaki. Optimal branch-decomposition of planar graphs in $O(n^3)$ time. *ACM Transactions on Algorithms*, 4(3), 2008.

[144] M. Hajiaghayi. *The bidimensionality theory and its algorithmic applications.* PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2005.

[145] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[146] F. Havet. Channel Assignment and Multicoloring of the Induced Subgraphs of the Triangular Lattice. *Discrete Mathematics*, 233(1-3):219–231, 2001.

[147] R. B. Hayward and R. Shamir. A note on tolerance graph recognition. *Discrete Applied Mathematics*, 143(1-3):307–311, 2004.

[148] M. R. Henzinger, S. Rao, and H. N. Gabow. Computing Vertex Connectivity: New Bounds from Old Techniques. *Journal of Algorithms*, 34(2):222–250, 2000.

[149] I. Holyer. The NP-Completeness of Some Edge-Partition Problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[150] Y. Hong-Hsu, S. Lee, and B. Mukherjee. Traffic grooming and delay constrained multicast routing in IP over WDM networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 5246–5251, 2008.

[151] J. Hu. Traffic Grooming in WDM Ring Networks: A Linear Programming Solution. *OSA Journal of Optical Networks*, 1(11):397–408, 2002.

[152] S. Huang, R. Dutta, and G. Rouskas. Traffic Grooming in Path, Star, and Tree Networks: Complexity, Bounds, and Algorithms. *IEEE Journal on Selected Areas in Communications*, 24(4):66–82, 2006.

[153] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989.

[154] R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. Special issue of FOCS 1998.

[155] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35, 1991.

[156] D. Karger, R. Motwani, and G. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997.

[157] M. Kaufmann, J. Kratochvíl, K. A. Lehmann, and A. R. Subramanian. Max-tolerance graphs as intersection graphs: cliques, cycles, and recognition. In *Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 832–841, 2006.

[158] A. Kézdy. *Studies in Connectivity.* PhD thesis, University of Illinois at Urbana-Champaign, 1991.

[159] S. Khot. Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[160] P. N. Klein, R. Krishnan, B. Raghavachari, and R. Ravi. Approximation Algorithms for Finding Low-Degree Subgraphs. *Networks*, 44(3):203–215, 2004.

[161] M. Köhn. A New Efficient Online-Optimization Approach for SDH/SONET-WDM Multi Layer Networks. In *Proceedings of Optical Fiber Communication Conference (OFC)*, 2006.

[162] G. Kreweras. Sur les partitions non croisées d'un cercle. *Discrete Mathematics*, 1:333–350, 1972.

[163] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Oxford University Press, 1976.

[164] L. Liu, X. Li, P.-J. Wan, and O. Frieder. Wavelength Assignment in WDM Rings to Minimize SONET ADMs. In *Proceedings of IEEE INFOCOM*, pages 1020–1025, 2000.

[165] L. Lovász and M. Plummer. *Matching Theory.* Annals of Discrete Mathematics 29, North-Holland, 1986.

[166] C. Lund and M. Yannakakis. The Approximation of Maximum Subgraph Problems. *Proceedings of the 20th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 40–51, volume 700 of LNCS, 1993.

[167] L. Mathieson, E. Prieto, and P. Shaw. Packing edge disjoint triangles: A parameterized view. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 127–137, volume 3162 of LNCS, 2004.

[168] L. Mathieson and S. Szeider. The Parameterized Complexity of Regular Subgraph Problems and Generalizations. In *Proceedings of the 14th Computing: The Australasian Theory Symposium (CATS)*, pages 79–86, volume 77 of CRPIT, 2008.

[169] C. McDiarmid and B. Reed. Channel Assignment and Weighted Colouring. *Networks*, 36(2):114–117, 2000.

[170] E. Modiano and P. Lin. Traffic grooming in WDM networks. *IEEE Communications Magazine*, 39(7):124–129, 2001.

[171] B. Mohar and C. Thomassen. *Graphs on surfaces.* John Hopkins University Press, 2001.

[172] H. Moser and D. M. Thilikos. Parameterized Complexity of Finding Regular Induced Subgraphs. *Journal of Discrete Algorithms*, 7(2):181–190, 2009.

[173] J. Munro and V. Raman. Succinct Representation of Balanced Parentheses and Static Trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[174] R. Niedermeier. *Invitation to Fixed Parameter Algorithms.* Oxford University Press, 2006.

[175] F. G. Nocetti, I. Stojmenović, and J. Zhang. Addressing and Routing in Hexagonal Networks with Applications for Tracking Mobile Users and Connection Rerouting in Cellular Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):963–971, 2002.

[176] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

[177] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[178] J. P. Petersen. Die Theorie der Regulären Graphs. *Acta Mathematica*, 15:193–220, 1891.

[179] M. Plantholt. The chromatic index of graphs with a spanning star. *Journal of Graph Theory*, 5:45–53, 1981.

[180] G. Quattrocchi. Embedding path designs in 4-cycle systems. *Discrete Mathematics*, 255:349–356, 2002.

[181] R. Ravi, M. Marathe, S. Ravi, D. Rosenkrantz, and H. Hunt III. Approximation Algorithms for Degree-Constrained Minimum-Cost Network-Design Problems. *Algorithmica*, 31(1):58–78, 2001.

[182] O. Richard. The Number of Trees. *Annals of Mathematics*, Second Series 49(3):583–599, 1948.

[183] N. Robertson and P. Seymour. Graph Minors X. Obstructions to Tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[184] N. Robertson and P. Seymour. Graph Minors XII. Distance on a Surface. *Journal of Combinatorial Theory, Series B*, 64:240–272, 1995.

[185] N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994.

[186] N. Robertson and P. D. Seymour. Graph Minors XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[187] N. Robertson and P. D. Seymour. Graph Minors XVI. Excluding a Non-Planar Graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003.

[188] N. Robertson and P. D. Seymour. Graph Minors XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[189] A. Rosa and W. D. Wallis. Premature sets of 1-factors, or, how not to schedule round-robin tournaments. *Discrete Applied Mathematics*, 4:291–297, 1982.

[190] P. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[191] M. Shalom, W. Unger, and S. Zaks. On the Complexity of the Traffic Grooming Problem in Optical Networks. In *Proceedings of the 4th International Conference on Fun with Algorithms*, pages 262–271, volume 4475 of LNCS, 2007.

[192] M. Shalom and S. Zaks. A $10/7 + \epsilon$ approximation scheme for minimizing the number of ADMs in SONET rings. In *Proceedings of BROADNETS*, pages 254–262, 2004.

[193] Y. Shiloach. Another look at the degree constrained subgraph problem. *Information Processing Letters*, 12(2):89–92, 1981.

[194] A. Somani. *Survivability & Traffic Grooming in WDM Optical Networks*. Cambridge Press, 2006.

[195] I. A. Stewart. Finding Regular Subgraphs in Both Arbitrary and Planar Graphs. *Discrete Applied Mathematics*, 68(3):223–235, 1996.

[196] D. H. Su and D. W. Griffith. Standards activities for MPLS over WDM networks. *Optical Networks Magazine*, 1(3), 2000.

[197] K. S. Sudeep and S. Vishwanathan. A technique for multicoloring triangle-free hexagonal graphs. *Discrete Mathematics*, 300(1-3):256–259, 2005.

[198] A. Suzuki and T. Tokuyama. Dense subgraph problem revisited. In *Proceedings of the Joint Workshop "New Horizons in Computing" and "Statistical Mechanical Approach to Probabilistic Information Processing"*, 2005.

[199] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

[200] C. Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989.

[201] C. Thomassen. Two-coloring the edges of a cubic graph such that each monochromatic component is a path of length at most 5. *Journal of Combinatorial Theory, Series B*, 75(1):100–109, 1999.

[202] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.

[203] P.-J. Wan, G. Calinescu, L. Liu, and O. Frieder. Grooming of arbitrary traffic in SONET/WDM BLSRs. *IEEE Journal of Selected Areas in Communications*, 18(10):1995–2003, 2000.

[204] J. Wang, W. Cho, V. R. Vemuri, and B. Mukherjee. Improved approaches for cost-effective traffic grooming in WDM ring networks: ILP formulations and single-hop and multihop connections. *IEEE/OSA Journal of Lightwave Technology*, 19(11):1645–1653, 2001.

[205] R. M. Wilson. Decomposition of complete graphs into subgraphs isomorphic to a given graph. *Congressus numerantium*, 15:647–659, 1976.

[206] S. Win. On a Connection Between the Existence of k-Trees and the Toughness of a Graph. *Graphs and Combinatorics*, 5(1):201–205, 1989.

[207] E. B. Yavorskii. Representations of directed graphs and ψ-transformations. *Theoretical and Applied Questions of Differential Equations and Algebra*, pages 247–250, 1978. A. N. Sharkovskii (Editor).

[208] X. Zhang and C. Qiao. An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings. *IEEE/ACM Transactions on Networking*, 8(5):608–617, 2000.

[209] K. Zhu and B. Mukherjee. A review of traffic grooming in WDM optical networks: Architectures and challenges. *Optical Networks Magazine*, 4(2):55–64, 2003.

[210] K. Zhu, H. Zhu, and B. Mukherjee. Traffic engineering in multigranularity heterogeneous optical WDM mesh networks through dynamic traffic grooming. *IEEE Network*, 17(2):8–15, 2003.

[211] K. Zhu, H. Zhu, and B. Mukherjee. *Traffic Grooming in Optical WDM Mesh Networks*. Springer, 2005.