Minimizing the number of ADMs in WDM Optical Rings with Traffic Grooming

PROJECTE FI DE CARRERA
ETSETB

Ignasi Sau Valls

Director: Xavier Muñoz López

Departament de Matemàtica Aplicada IV
Universitat Politècnica de Catalunya

Als meus pares.

Acknowledgements

First of all, I would like to thank my parents sincerely the constant support that they give to me day after day. I know that I should express much more often this feeling to them, but I am convinced that I would not have done anything without the surety that they would do their best to help me if something went wrong.

I am very pleased to give all my gratitude to Dr. Xavier Muñoz for all the opportunities that I have had because of him. This work begun with him at *Departament de Matemàtica Aplicada IV* of *UPC*, and then I had the great change to follow my work at *INRIA* in *Sophia Antipolis* (Nice) from September until March, 2006. There I met Dr. Jean-Claude Bermond and Dr. David Coudert, to whom I have to thank all the things that I learned and the fabolous atmosphere that there is in *MASCOTTE* team. I am really happy to have started my PhD under the shared supervision of they three.

I have to thank the member staff of *CFIS* for having carried out successfully this ambitious project. Above all, Prof. Josep Grané, Prof. Marta València and Prof. Pere Pascual.

I cannot forget my friends in the *Faculty of Mathematics* of *UPC* : Bird, Chous, J, Pino, Txema, Pil, Molina, Max, Ricard, Roger, Pau, Uri, Laura, Marcel, Carlos, XSortas and Pep, among many others. The huge amount of hours that we have spent together working (*et al.*..) during the last 6 years makes me feel that they have become my second family and that *FME* is my second home. Of course, there would be no home without a door, and without people in charge of opening this door every day (and night), always with a smile on their face : Jaime, Jorge and Albert.

The friends that I met at *INRIA* and Nice during the time I spent there also deserve my deepest memory : Ivan, Ernesto, Pepe, Edu, Torsten, Florian, Omid, Hervé, Caro, Vero, Palo, Albert, María, Helena, Malena, Giova, Faouzi, Abdulhalim, Ozge, Pinar, Tirema, Luc, Pato, Nelson, Fred, Stephan and Michel, to cite some of them.

I absolutey want to remember here my childhood friends. Although we do not spend enough time together (it is true, I am always abroad...), I know that nothing will change in our friendship : Carlos, David, Luis, Cristian, Meju, Rus, Sergi, Ángel...

Finally, I have no words for Mar, probably the most special person I have ever met.

Resum

El "traffic grooming" en xarxes òptiques es refereix a agrupar tràfic de baixa velocitats dintre de streams de tràfic més ràpids. Fent servir "traffic grooming" es pot prescindir d'equipament electrònic als nodes de la xarxa on no comença ni acaba tràfic, i per tant es pot reduir el cost de la xarxa. Típicament, en una xarxa WDM (Wavelength Division Multiplexing), enlloc de tenir un SONET Add-Drop Multiplexer (ADM) a cada longitud d'ona i a cada node, és possible tenir ADMs només per les longituds d'ona utilitzades a cada node (la resta de longituds d'ona són enrutades òpticament sense conversió electrònica). Hi ha moltes variants segons el tipus de xarxa (anell, arbre, camí...), les restriccions que es tenen i els paràmetres que es volen optimitzar.

L'objectiu és minimitzar el cost d'equipament de la xarxa, i concretament el número d'ADMs. Aquest problema ha estat àmpliament estudiat durant els últims anys. A grans trets, hi ha dues línies d'atac. D'una banda, hi ha el camp de les heurístiques i els algorismes aproximatius. D'altra banda, es poden aplicar mètodes més teòrics de la teoria de grafs per trobar solucions òptimes amb el mínim número d'ADMs. Aquest últim és el nostre mètode. Hi ha una àmplia varietat de topologies a considerar. S'han fet heurístiques i algorismes per gairebé tots els tipus de xarxa, però en canvi només s'han trobat solucions òptimes per l'anell unidireccional i el camí.

El problema de "traffic grooming" es pot traduir a un problema de descomposició de grafs, on s'apliquen eines de teoria de dissenys i de teoria de grafs. Una lleugera variació en l'enrutament pot canviar completament el problema, i hi ha un constant compromís entre la simplicitat de tractament i el valor pràctic de cada tipus d'enrutament (simètric, camí més curt...).

A la primera part d'aquest Projecte Fi de Carrera, estudiem la minimització d'ADMs en anells òptics WDM bidireccionals amb enrutament simètric pel camí més curt i amb requeriments de tràfics unitaris i tots-amb-tots. Insistim en el plantejament del problema, que no havia estat plantejat amb rigor en el cas bidireccional. Encara no s'havien trobat solucions òptimes. Nosaltres, en particular estudiem els casos C=2 i C=3 (sent C el "grooming factor") trobant construccions òptimes o quasi-òptimes. També estudiem el cas C=k(k+1)/2 trobant descomposicions òptimes per famílies infinites de valors de N (número de nodes de la xarxa). Establim una fita inferior general, i la millorem per C=2,3. També incloem algun resultat per valors grans de C i comentaris sobre la formulació del problema en termes de Programació Lineal. Al final d'aquesta primera part donem alguns resultats quan el graf de requeriments és circulant.

A la segona part d'aquest treball en concentrem en variants del problema a l'anell unidireccional. Primer considerem el problema clàssic però canviant el conjunt de requeriments. Concretament, trobem solucions òptimes per a alguns casos quan el graf de requeriments té grau màxim fitat. Finalment, considerem una modificació del problema que és una primera aproximació al grooming dinàmic des d'un punt de vista purament teòric. Extenem les idees d'un article que està a punt de publicar-se, i trobem noves solucions òptimes.

Abstract

Traffic Grooming in optical networks refers to group low rate traffic into higher speed streams. By using traffic grooming one can bypass the electronics in the nodes for which there is no traffic sourced or destinated to it and therefore reduce the cost of the network. Typically, in a WDM (Wavelength Division Multiplexing) network, instead of having one SONET Add-Drop Multiplexer (ADM) on every wavelength at every node, it may be possible to have ADMs only for the wavelength used at that node (the other wavelengths being optically routed without electronic switching). There are many variants according to the type of network considered (for example, path, ring or tree), the constraints used and the parameters one wants to optimize which give rise to a lot of interesting design problems (graph decomposition).

The objective is to minimize the equipment cost of the network, and specifically the number of ADMs. This problem has been widely studied by many researchers in the last few years. Roughly, there are two lines of attack. On the one hand, there is the field of heuristics and approximation algorithms. On the other hand, more theoretical methods from graph theory can be applied to find optimal solutions with the minimum number of ADMs. This last one is our approach. There is a wide variety of topologies to be considered. Heuristics and algorithms have been found for almost all type of networks, but it is not the case of exact solutions at all, where only the unidirectional ring and the path have been properly studied. The problem of traffic grooming can be translated to a problem of graph partitioning, where tools from graph theory and design theory are strongly needed. A slight variation on the routing may completely change the problem, and there is a constant trade-off between the simplicity of the formulation and the practical value of each type of routing (symmetric, shortest path...).

In the first part of this Master Thesis, we study the minimization of ADMs in Optical WDM Networks with Bidirectional Ring topology considering symmetric shortest path routing and all-to-all unitary requests. We insist on the statement of the problem, which had not been clearly stated before in the bidirectional case. Furthermore, optimal solutions had not been found up to date.
The *grooming factor* es denoted by C. In particular, we study the case C = 2 and C=3 (giving either optimal constructions or near-optimal solutions) and the case C = k(k+1)/2 (giving optimal decompositions for specific congruence classes of N). We state a general Lower Bound for all the values of C and N, and we improve this Lower Bound for C=2 and C=3 (when N=4t+3). We also include approaches to large values of C and some comments about the formulation of the problem using Linear Programming.

At the end of this first part we give some results when the set of requests consists in a circulant graph.

In the second part of this dissertation we focus on variants of the problem in the unidirectional ring. First of all, we consider the classical problem but changing the set of requests (optimal solutions had been found in the all-to-all case). Specifically, we find optimal solutions for some cases when the set of requests consists of graphs with bounded degree.
Finally, we consider a modification of the problem that is a first approach to the dynamic grooming from a purely theoretical point of view. We extend the ideas from an article that is going to appear soon, and we find new optimal solutions.

Table of Contents

List of Figures

List of Tables

Nomenclature

In the next pages we list the notation that we will refer along the Master Thesis. The list is done following the order of appearance, and not taking into account the Section of preliminary notions.

$\mathbf{G}$	Network or physical graph.
$\mathbf{V(G)}$	Vertex set of graph G.
$\mathbf{E(G)}$	Edge set of graph G.
$\mathbf{N}$	Number of nodes of the network.
$\overrightarrow{C}_{\mathbf{N}}$	Unidirectional ring on N nodes.
$\mathbf{C_N^*}$	Bidirectional ring on N nodes.
$\mathbf{I}$	Request or logical graph.
$\mathbf{K_N}$	Complete graph on N nodes.
$\mathbf{P(r)}$	Path assigned to request r.
$\mathbf{(i,j)}$	Request between nodes i and j.
$\mathbf{\{i,j\}}$	Request between nodes i and j.
$\mathbf{C}$	Grooming factor or grooming ratio.
$\mathbf{B}_\omega$	Set of requests on wavelength ω.
$\mathbf{L(B_\omega,e)}$	Load at edge e on wavelength ω.
$\mathbf{A_r}$	Set of allowed paths in G for request r.
$\mathbf{A}$	$\bigcup_{r\in I} A_r$.
$\mathbf{A(G,I,A,C)}$	Optimal solution of the Traffic Grooming problem, given G, I, A and C.
$\mathbf{T_N}$	Tournament graph on N nodes.
$\mathbf{A(C,N)}$	Optimal solution of the Traffic Grooming problem, given C and N, and assuming symmetric shortest path routing and all-to-all unitary requests.
$\mathbf{G-e}$	Graph obtained from G by deleting an edge.
$\mathbf{a_p}$	Number of graphs with p vertices.
$\mathbf{A}$	Number of ADMs in a generic solution.
$\mathbf{W}$	Number of wavelengths in a generic solution.
$\mathbf{V}_\omega$	$E(V_\omega)$.
$\mathbf{E}_\omega$	$E(B_\omega)$.
$\gamma(\mathbf{C},\mathbf{p})$	Maximum number of edges of any graph $H=(V,E)$ with $\lvert V\rvert = p$, such that $L(H,e)\le C$, $\forall e\in E(H)$.
$\mathbf{l_G}$	Length of a request in the graph G.
$\mathbf{l_I}$	Length of a request in a subgraph B_ω of I.
$\rho(\mathbf{B}_\omega)$	$\frac{\lvert V(B_\omega)\rvert}{\lvert E(B_\omega)\rvert}$.
$\rho_{\mathbf{min}}(\mathbf{C})$	$\min\{\rho(B_\omega) \mid L(B_\omega,e)\le C\ \forall e\in E(B_\omega)\}$.
$\mathbf{R}$	Number of requests in a generic solution.

$\mathbf{d_{jk}}$	Amount of traffic demands from node i to node j.
$\widehat{\mathbf{L}}(\mathbf{r},\mathbf{B}_\omega)$	$\max_{e\in G} L(e, B_\omega - r)$.
$\mathbf{c}(\mathbf{v},\mathbf{l},\mathbf{t})$	Minimum number of blocks in any $t-(v,l,\lambda)$ covering.
$\rho(\mathbf{n})$	Minimum number of cycles in a DRC-covering of K_n.
$\mathbf{P_N}$	Path on N nodes.
$\mathbf{K_{s\times r}}$	Complete r-partite graph $\overline{K_{n_1}} * \ldots * \overline{K_{n_r}}$, with $n_1 = \ldots = n_r =: s$.
BIBD	Balanced Incomplete Block Design.
$\overline{\delta}$	Average degree of a graph.
$\boldsymbol{\Delta}$	Maximum degree of a graph.
δ	Minimum degree of a graph.

In Grooming for Two-Period Optical Networks (Chapter 4) :

$\mathbf{C}$	Grooming factor for the first period.
$\mathbf{C'}$	Grooming factor for the second period.
$\mathbf{X}$	Set of nodes of the ring. $\|V\| = n$.
$\mathbf{V}$	Subset of V for which C' applies.$\|V'\| = v$.
$\mathbf{W}$	$X \setminus V$.
$\mathbf{N}(\mathbf{n},\mathbf{C})$	Grooming in a ring on n vertices and grooming ratio C.
$\mathscr{ON}(\mathbf{n},\mathbf{C})$	Optimal $N(n,C)$.
cost $\mathbf{N}(\mathbf{n},\mathbf{C})$	Drop cost of $N(n,C)$.
cost $\mathscr{ON}(\mathbf{n},\mathbf{C})$	Drop cost of $\mathscr{ON}(n,C)$.
$\mathbf{N}(\mathbf{n},\mathbf{v};\mathbf{C},\mathbf{C'})$	Grooming in a two-period network.
$\mathscr{ON}(\mathbf{n},\mathbf{v};\mathbf{C},\mathbf{C'})$	Optical grooming in a two-period network.
cost $\mathbf{N}(\mathbf{n},\mathbf{v};\mathbf{C},\mathbf{C'})$	Drop cost of $N(n,v;C,C')$.
cost $\mathscr{ON}(\mathbf{n},\mathbf{v};\mathbf{C},\mathbf{C'})$	Drop cost of $\mathscr{ON}(n,v;C,C')$.
$\aleph(\mathbf{n},\mathbf{C},\mathbf{C'})$	Set of integers for wich *cost* $\mathscr{ON}(n,v;C,C') =$ *cost* $\mathscr{ON}(n,C)$.

Motivation

What is traffic grooming ?

Traffic grooming in WDM networks is defined as the allocation of sub-wavelength traffic tributaries onto full wavelengths channels in order to achieve efficient utilization of network resources, such that some cost function be minimized. This is usually implemented according to a strategy that optimizes a certain objective function, such as minimizing cost or blocking probability, or maximizing revenue.

Realization that most of the applications' bandwidth requirements are sub-wavelength has put the traffic grooming under the spotlight, and increased its importance. In the beginning, the motivation for traffic grooming was merely reducing the total number of required wavelengths, such that, given limited number of wavelengths, maximum amount of traffic can be accommodated.

However, the understanding that traffic grooming can significantly reduce the number of higher layer components, and thus network cost, has created an enormous interest in this area. Starting with the consideration of regular topologies and specific static traffic patterns, this area has seen many advances in just the last few years.

Outline of this Master Thesis

This Master Thesis is structured as follows. In Chapter 1 we sum all the preliminary concepts needed for reading this work, and we present the problem of Traffic Grooming. We state the general version of the problem and we describe briefly the situation of the state-of-the-art. Finally, we talk a bit about the complexity of the problem.

Chapter 2 is the core of this dissertation. We focus on the Bidirectional Ring Grooming Problem with all-to-all unitary requests and shortest path symmetric routing. Although it may seem a very restricted case, its applications in real networks make it really interesting. We found new results that yield some optimal solutions and approximations.

In Chapter 3 we study other particular cases of the general Traffic Grooming Problem. First of all, we consider the unidirectional ring with a set of requests made up of graphs with a bounded degree. It is natural to study this scenario because the amount of traffic generated at each node cannot be arbitrarily large in real networks.
In the second part we focus on the bidirectional ring case, with circulant graphs as set of

requests. This is also a natural case to be considered, because frequently a node only has traffic requirements with its nearest nodes of the network, due to distance or reachability constraints, for example.

In Chapter 4 we deal with a variation of the Traffic Grooming Problem : the Grooming for Two-Period Optical Networks. In this case there are two possible values of the grooming factor, that affect different subsets of the node set of the network.
It is a first approach to the dynamic grooming using graph partitioning tools.

Finally, we give general conclusions and future work to be done in this area.

Research Contributions

Fruit of the work carried out doing this Master Thesis, we have already presented a poster and a paper, and we are still finishing two other articles :

- Poster :

  Traffic Grooming in Optical Networks.
  ADONET/COST293 Spring School on Combinatorial Optimization and Communication Networks, Budapest University of Technology and Economics 20th-24th 3-2006.

- Paper :

  Jean-Claude Bermond, David Coudert, Xavier Muñoz and Ignasi Sau. **Traffic Grooming in Bidirectional WDM Ring Networks**. In *IEEE/LEOS/COST 293 annual conference of GRAAL*, which forms part of *8th ICTON, Nottingham, UK, June 2006.*

- 2 articles in preparation.

The poster is provided in Appendix B, and the paper in Appendix C.

Chapter 1

Introduction

Abstract

In this Chapter we provide all the necessary concepts that will be used along this Master Thesis. We will not pretend that the reader reads all the definitions carefully, this Chapter is conceived as being only for consulting. In the second part we formally state the Traffic Grooming problem and we make a fast overview of the state-of-the-art.

1.1 Preliminary notions

Some information of all this Section has been taken in part from [1, 2]. Other sources of information will be specified in each case.

1.1.1 Optical Networks : SONET/SDH

SONET and SDH are a set of related standards for synchronous data transmission over fiber optic networks. SONET is short for Synchronous Optical NETwork and SDH is an acronym for Synchronous Digital Hierarchy. SONET is the United States version of the standard published by the American National Standards Institute (ANSI). SDH is the international version of the standard published by the International Telecommunications Union (ITU).

The SONET/SDH Digital Hierarchy

Table 1.1 lists the hierarchy of the most common SONET/SDH data rates.

Optical Level	*Electrical Level*	*Line Rate*	*Payload R.*	*Overhead R.*	*SDH*
$OC-1$	$STS-1$	51.840	50.112	1.728	–
$OC-3$	$STS-3$	155.520	150.336	5.184	$STM-1$
$OC-12$	$STS-12$	622.080	601.344	20.736	$STM-4$
$OC-48$	$STS-48$	2488.320	2405.376	82.944	$STM-16$
$OC-192$	$STS-192$	9953.280	9621.504	331.776	$STM-64$
$OC-768$	$STS-768$	39813.120	38486.016	1327.104	$STM-256$

Tab. 1.1 – Hierarchy of the most common SONET/SDH data rates (in Mbps)

Other rates (OC-9, OC-18, OC-24, OC-36, OC-96) are referenced in some of the standards documents but were never widely implemented. It is possible that other higher rates (e.g. OC-3072) may be defined in in the future. We can see an scheme of SONET signal mappings in Figure 1.1.

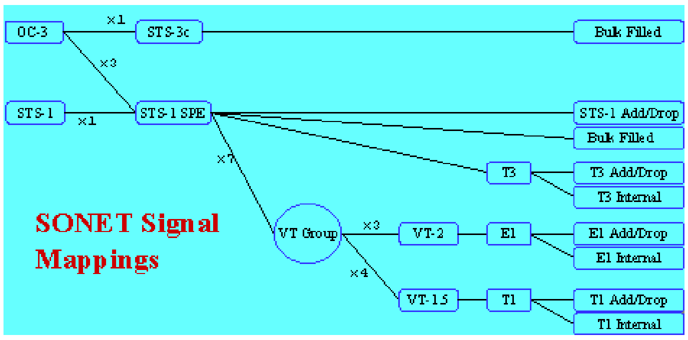


Fig. 1.1 – SONET Signal Mappings

The "line rate" refers to the raw bit rate carried over the optical fiber. A portion of the bits transferred over the line are designated as "overhead". The overhead carries information that provides OAM&P (Operations, Administration, Maintenance, and Provisioning) capabilities such as framing, multiplexing, status, trace, and performance monitoring. The "line rate" minus the "overhead rate" yields the "payload rate" which is the bandwidth available for transferring user data such as packets or ATM cells.

The SONET/SDH level designations sometimes include a "c" suffix (such as "OC-48c"). The "c" suffix indicates a "concatenated" or "clear" channel. This implies that the entire payload rate is available as a single channel of communications (i.e. the entire payload rate may be used by a single flow of cells or packets). The opposite of concatenated or clear channel is "channelized". In a channelized link the payload rate is subdivided into multiple fixed rate channels. For example, the payload of an OC-48 link may be subdivided into four OC-12 channels. In this case the data rate of a single cell or packet flow is limited by the bandwidth of an individual channel.

SONET/SDH Digital Standards

The American National Standards Institute (ANSI) coordinates and approves SONET standards. The standards are actually developed by Committee T1 which is sponsored by the Alliance for Telecommunications Industry Solutions (ATIS) and accredited by ANSI to create network interconnection and interoperability standards for the United States. T1X1 and T1M1 are the primary T1 Technical Subcommittees responsible for SONET. T1X1 deals with "digital hierarchy and synchronization". T1M1 deals with "internetworking operations, administration, maintenance, and provisioning" (OAM&P). Refer to the ANSI web site at *http ://www.ansi.org* for a complete list of SONET standards.

The International Telecommunications Union (ITU) coordinates the development of SDH standards. ITU was formerly known as the CCITT. It is sponsored by the United Nations and coordinates the development of telecommunications standards for the entire world. Refer to the ITU web site at *http ://www.itu.int* for a complete list of SDH standards.

Brief SONET Glossary

- **WDM - Wavelength Division Multiplexing**. WDM takes optical signals (each carrying information at a certain bit rate), gives them a color (a wavelength or specific frequency), and then sends them down the same fiber (see Figure 1.2). Several wavelengths are multiplexed in the same fiber. Typically, each wavelength uses 40Gbps, and the fiber capacity is in the Tbps, so hundreds of wavelengths can be multiplexed in each fiber.

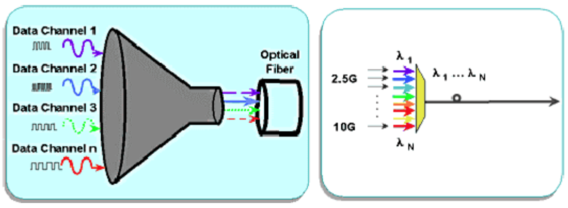


FIG. 1.2 – Multiplexing Optical Signals (WDM)

- **TDM - Time Division Multiplexing**. At each wavelength of each fiber, time division multiplexing of the STM-n signals is done. $OC-N$ means that we put N STM signals in a single wavelength. That's the idea of **grooming**. For example, if the electrical signals are $OC-12$ and a wavelength can carry an $OC-48$, then the *grooming factor* is 4 (see Figure 1.3).
  Thus, TDM is done first, and then WDM, increasing in this way the capacity of the network. We can see a scheme of the relation between WDM and TDM in Figure 1.4.
  More precisely, SONET TDM takes synchronous and asynchronous signals and multiplexes them to a single higher bit rate for transmission at a single wavelength over fiber. Source signals may have to be converted from electrical to optical, or from optical to electrical and back to optical before being multiplexed. WDM takes multiple optical signals, maps them to individual wavelengths, and multiplexes the wavelengths over a single fiber. Another fundamental difference between the two technologies is that WDM can carry multiple protocols without a common signal format, while SONET cannot. Some of the key differences between TDM and WDM are graphically illustrated in Figure 1.5.

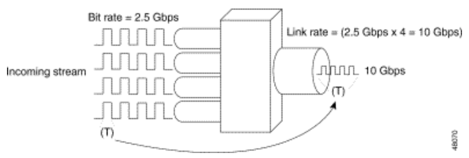


FIG. 1.3 – SONET TDM with grooming ratio 4

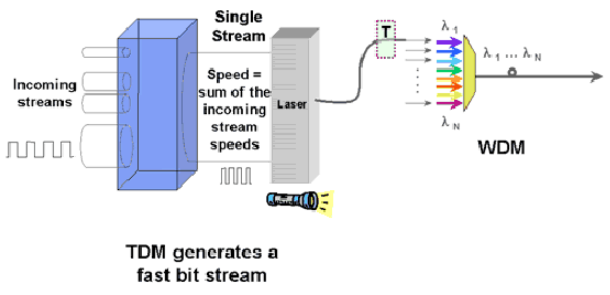

Incoming streams
Single Stream
Speed = sum of the incoming stream speeds
Laser
T
λ_1
$\lambda_1 \ldots \lambda_N$
λ_N
WDM
TDM generates a fast bit stream

FIG. 1.4 – Relation between TDM and WDM

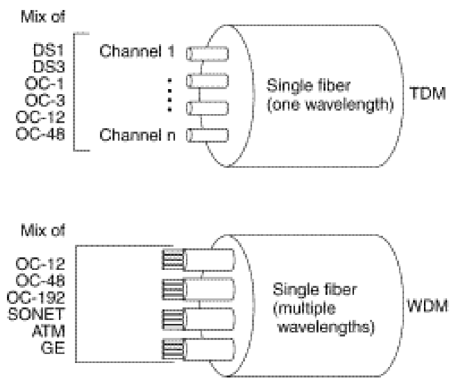


FIG. 1.5 – TDM and WDM Interfaces

- **F-OXC - Fiber Optical CrossConnect**. It does fiber switching with λ (wavelength) adding and dropping. An OADM is needed for each fiber from which λ adding and dropping is done. If there exists B-OXC, then F-OXC does band adding and dropping. We can see a scheme of a F-OXC in Figure 1.6.

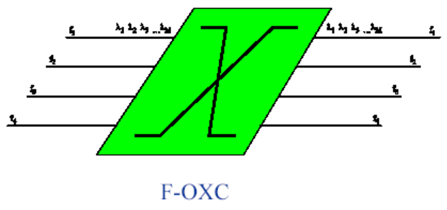


FIG. 1.6 – Scheme of a F-OXC

- **B-OXC - Band Optical CrossConnect**. It does band switching with λ adding and dropping. As before, an OADM is needed for each fiber from which λ adding and dropping is done. We can see a scheme of optical wavebands in Figure 1.7.

- **W-OXC - Wavelength Optical CrossConnect**. It does wavelength switching with (WT-OXC) or without (WR-OXC) λ conversion, and with STM electrical signals adding and dropping. Typically, the electrical signals are STM-1 (155.52Mbps). An ADM is needed for each wavelength from which STM adding and dropping is

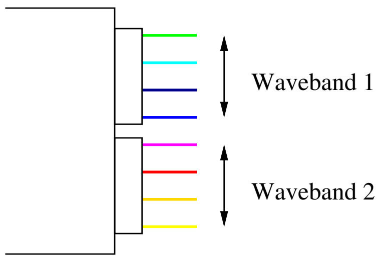


FIG. 1.7 – Scheme of wavebands

done. We can see a block scheme of a typical optical network in Figure 1.8.

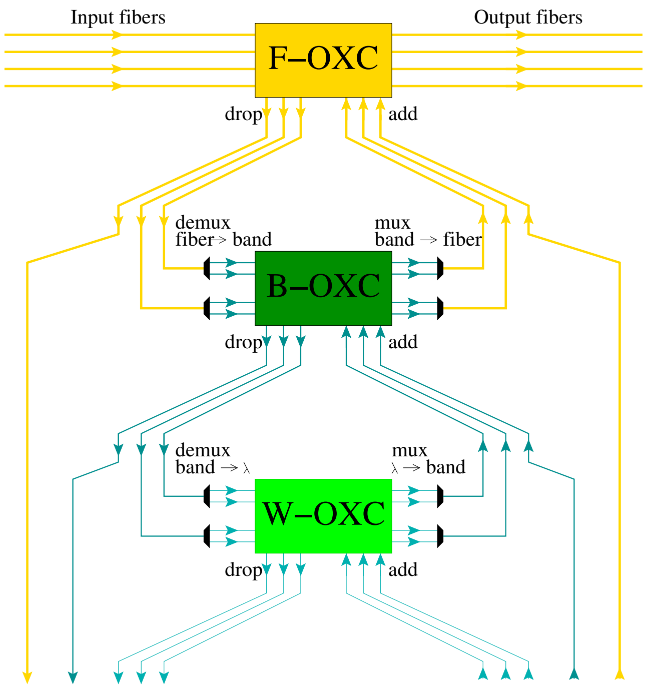


FIG. 1.8 – Blocks of a typical optical fiber architecture

- **OADM - Optical Add-Drop Multiplexer**. It does λ adding and dropping from a fiber. We can see a scheme of an Optical Add-Drop Multiplexer in Figure 1.9.

- **ADM - Add-Drop Multiplexer**. It does electrical signals adding and dropping from a wavelength. This is one of SONET's claim to fame. The tributaries of a SONET transport stream, are synchronously multiplexed to the line rate, i.e. there are no stuff bits or stuff opportunity bits as is the case in the plesiochronous hierarchy. As such an ADM can insert or extract lower rate tributary data without demultiplexing the aggregate line rate. We can see a block scheme of an Add-Drop Multiplexer in Figure 1.10, and both ADM/OADM in Figure 1.11.

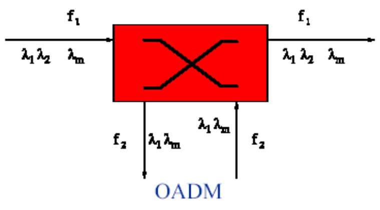


FIG. 1.9 – Scheme of an OADM

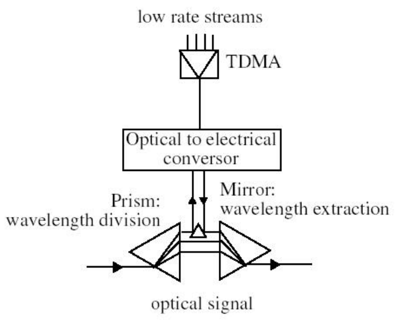

low rate streams
TDMA
Optical to electrical conversor
Prism: wavelength division
Mirror: wavelength extraction
optical signal

FIG. 1.10 – Block Scheme of an Add-Drop Multiplexer

1.1.2 Graph Theory

Additional references for this Section can be found in [19, 23].

Definition 1.1.1 (Digraph) *A* directed graph*, or, simply, a* digraph *is a pair $G = (V, E)$ of sets $V(G)$ and $E(G)$, such that $E(G) \subseteq V(G) \times V(G)$. The elements of V are called* vertices *(or* nodes*, or* points*) of the graph G, the elements of E are its* arcs*. An arc of G or the form (x, x) is called a* loop*. If $e = (x, y)$ is an arc then x is its* initial vertex *and y its* end vertex*.*

Definition 1.1.2 (Symmetric digraph) *A directed graph is* symmetric *if (x, y) is an arc whenever (y, x) is.*

In many cases the distinction between initial and end vertices is irrelevant. Thus, the notion of *undirected graph* is introduced :

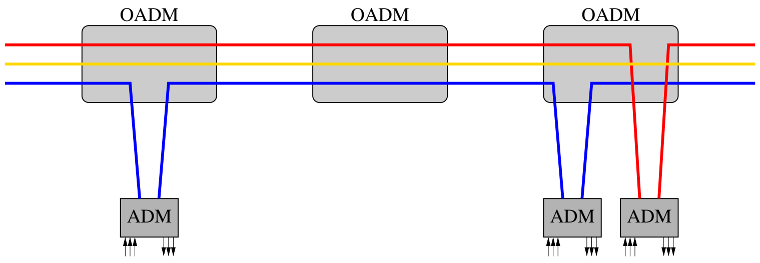


FIG. 1.11 – Block Scheme of both OADM and ADM

Definition 1.1.3 (Undirected Graph) *Compared to a digraph, in an* undirected graph *(or simply,* graph*) an arc (x,y) is replaced by the set consisting of the two vertices x and y, called and edge of the graph and denoted by $[x, y]$.*

A variation on the definition of digraph (or directed graph) is the *oriented graph*, which is a graph (or multigraph) with an orientation or direction assigned to each of its edges. For example (see Definition 1.1.15), by $\vec{C}$ we mean than only one direction is allowed for the arcs of the cycle.

A distinction between a directed graph and an oriented simple graph is that if x and y are vertices, a directed graph allows both (x, y) and (y, x) as edges, while only one is permitted in an oriented graph. Nevertheless, both concepts are used almost indistinctly.

Definition 1.1.4 (Multigraph) *When considering several edges with the same extremities we speak of a* multiple edge *and of a* multigraph*. If the (multi)graph has neither multiple edges nor loops it is called* simple*. By G^* we will denote the directed symmetric graph obtained from the graph G by replacing each edge $[x, y]$ by two arcs (x, y) and (y, x).*

The usual way to picture a graph is by drawing a dot for each vertex and joining two of these dots by a line (or a directed line if G is a digraph) if the corresponding two vertices form an edge (just how these dots and lines are drawn is considered irrelevant). We can see examples of graphs in Figure 1.12.

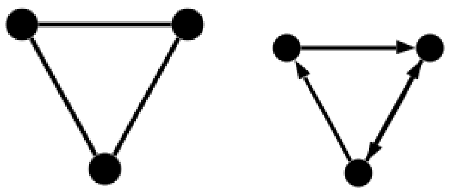


FIG. 1.12 – Examples of undirected and directed graphs

A graph with vertex set V is said to be a graph *on* V. Is is usual to speak of a vertex $v \in G$ (rather than $v \in V(G)$), an edge $e \in G$, and so on.

Definition 1.1.5 (Order) *The number of vertices of a graph G is its* order*, written as $|G|$.*

Graphs are *finite, infinite, countable* and so on according to their order. In all this work, we will deal with finite graphs.
For the *empty graph* $(\emptyset, \emptyset)$ we simply write $\emptyset$. A graph of order 0 or 1 is called *trivial.*

Definition 1.1.6 (Incidency) *A vertex v is* incident *with an edge e if $v \in e$; then e is an edge* at *v.*

The two vertices incident with and edge are its *endvertices* or *ends*, and an edge *joins* its ends. An edge $\{x, y\}$ is usually written as xy (or yx).
If $x \in X$ and $y \in Y$, than xy is an $X - Y$ *edge*. The set of all the edges in E at a vertex v is denoted by $E(v)$.

Definition 1.1.7 (Adjacency) *Two vertices x, y of G are* adjacent*, or* neighbours*, if xy is an edge of G. Two edges $e \neq f$ are adjacent if they have and end in common.*

Definition 1.1.8 (Complete graph) *If all the vertices of G are pairwise adjacent, then G is* complete*. A complete graph on* n *vertices is a K_n ; a K_3 is called a* triangle*.*

We can see in Figure 1.13 the complete graphs on $2, 3, 4, 5, 6$ and 7 vertices.

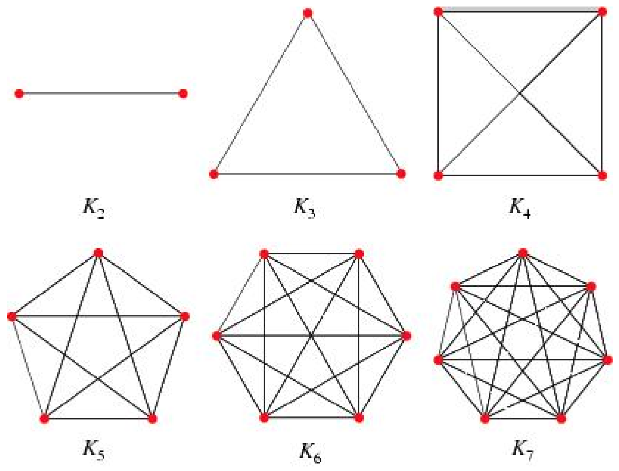


Fig. 1.13 – Some complete graphs

Definition 1.1.9 (Independence) *Pairwise non-adjacent vertices are called* independent*. More formally, a set of vertices or of edges is* independent *(or* stable*) if no two of its elements are adjacent.*

Definition 1.1.10 (Isomorphism) *Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. We call G and G'* isomorphic*, and write $G \simeq G'$, if there exists a bijection $\varphi : V \rightarrow V'$ with $xy \in E \Leftrightarrow \varphi(x)\varphi(y) \in E'$ for all $x, y \in V$. Such a map φ is called an* isomorphism *; if $G = G'$, it is called an* automorphism*.*

We do not normally distinguish between isomorphic graphs. Thus, we usually write $G = G'$ rather than $G \simeq G'$.
Let $G = (V, E)$ be a (non-empty) graph. The set of neighbours of a vertex v in G is denoted by $N_G(v)$, or briefly by $N(v)$. More generally for $U \subseteq V$, the neighbours in $V \setminus U$ of vertices in U are called *neighbours* of U ; their set is denoted by $N(U)$.
We set $G \cup G' := (V \cup V', E \cup E')$ and $G \cap G' := (V \cap V', E \cap E')$.

Definition 1.1.11 (Subgraph) *If $G \cap G' = \emptyset$, then G and G' are disjoint. If $V' \subseteq V$ and $E' \subseteq E$, then G' is a* subgraph *of G (and G a* supergraph *of G'), written as $G' \subseteq G$.*

Less formally, we say that G *contains* G'. If $G' \supseteq G$ and $G' \neq G$, then G' is a *proper subgraph* of G.
If $G' \subseteq G$ and G' and G' contains all the edges $xy \in V'$ with $x, y \in V'$, then G' is an *induced subgraph* of G ; we say that V' *induces* or *spans* G' in G, and write $G' =: G[V']$.

Definition 1.1.12 (Spanning subgraph) $G' = (V', E') \subseteq G = (V, E)$ *is a* spanning subgraph *of* G *if* V' *spans all of* G*, i.e. if* $V' = V$.

Definition 1.1.13 (Degree) *The* degree *(or* valency*)* $d_G(v) = d(v)$ *of a vertex* v *is the number* $|E(v)|$ *of edges at* v *; this is equal to the number of neighbours of* v*. A vertex of degree* 0 *is* isolated*. The number* $\delta(G) := min\{d(v)|v \in V\}$ *is the* minimum degree *of* G*, the number* $\Delta(G) := max\{d(v)|v \in V\}$ *its* maximum degree*.*

If all the vertices of G have the same degree k, then G is *k-regular*, or simply *regular*. A 3-regular graph is called *cubic*.

The number

$$d(G) := \frac{1}{|V|} \sum_{v \in V} d(v)$$

is the *average degree* of G. Clearly,

$$\delta(G) \leq d(G) \leq \Delta(G)$$

From these definitions the following properties are straightforward.

Proposition 1.1.1 *For all graph* $G = (V, E)$*,*

$$\sum_{v \in V} d(v) = 2|E|$$

Proposition 1.1.2 *The number of vertices of odd degree in a graph is always even.*

Definition 1.1.14 (Path) *A* path *is a non-empty graph* $P = (V, E)$ *of the form*

$$V = \{x_0, x_1, \ldots, x_k\} \quad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$$

where the x_i *are all distinct. The vertices* x_0 *and* x_k *are* linked *by* P *and are called its* ends *; the vertices* $x_1, \ldots, x_{k-1}$ *are the* inner *vertices of* P*. The number of edges of a path is its* length*, and the path of length* k *is denoted by* P^k*.*

Note that k is allowed to be zero ; thus $P^0 = K_1$. We can see an example of a path in Figure 1.14.

Definition 1.1.15 (Cycle) *A* cycle *is a non-empty graph* $P = (V, E)$ *of the form*

$$V = \{x_0, x_1, \ldots, x_k\} \quad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k, x_kx_0\}$$

where the x_i *are all distinct.*

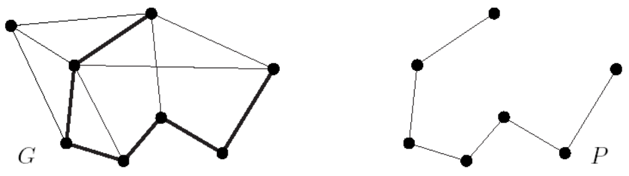


FIG. 1.14 – A path $P = P^6$ in G

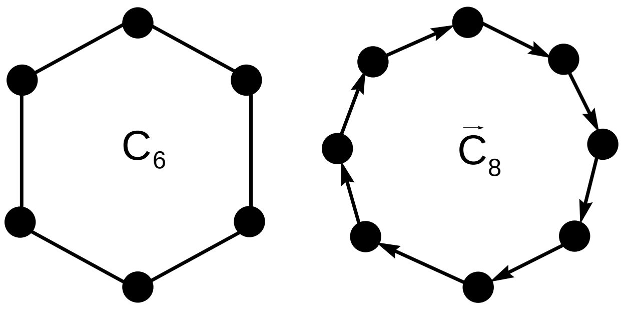


FIG. 1.15 – Examples of undirected and directed cycles

Note that if $P = x_0x_1 \dots x_{k-1}$ is a path and $k \geq 3$, then the graph $C := P + x_{k-1}x_0$ is a cycle. As with path, we often denote a cycle by its (cyclic) sequence of vertices. The *length* of a cycle is its number of edges (or vertices) ; the cycle of length k is called a $k-$*cycle* and denoted by C_k. A cycle, unlike a path, is not allowed to have length 0. We can see in Figure 1.15 examples of undirected and directed cycles.

Definition 1.1.16 (r-partite graph) *Let $r \geq 2$ be an integer. A graph $G = (V, E)$ is called* r-partite *if V admits a partition into r classes such that every edge has its ends in different classes : vertices in the same partition class must not be incident. Instead of '2-partite' one usually says* bipartite.

We can see an example of a bipartite graph in Figure 1.16. An r-partite graph in which

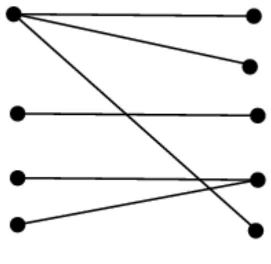


FIG. 1.16 – A bipartite graph

every two vertices from different partition classes are adjacent is called *complete*. We can see some examples of complete bipartite graphs in Figure 1.17, and a nice $K_{18,18}$ in Figure 1.18. The complete r-partite graphs for all r together are the *complete multipartite* graphs. The complete r-partite graph $\overline{K_{n_1}} * \ldots * \overline{K_{n_r}}$ is denoted by $K_{n_1,\ldots,n_r}$; if $n_1 = \ldots = n_r =: s$, we abbreviate this to K_s^r or $K_{s\times r}$.

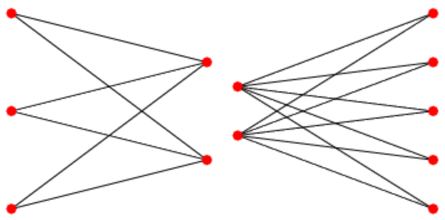


FIG. 1.17 – Some complete bipartite graphs

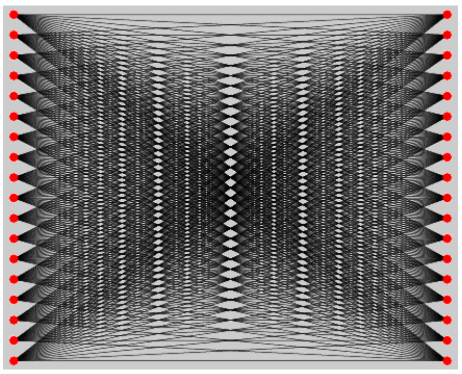


FIG. 1.18 – A nice $K_{18,18}$

Graphs of the form $K_{1,n}$ are called *stars* ; the vertex in the singleton partition class of this is the star's *center*.

Clearly, a bipartite graph cannot contain an *odd cycle*, a cycle of odd length. In fact, the bipartite graphs are characterized by this property :

Proposition 1.1.3 *A graph is bipartite if and only if it contains no odd cycle.*

Definition 1.1.17 (Girth, Circumference) *The minimum length of a cycle (contained) in a graph* G *is the* girth g(G) *of* G *; the maximum length of a cycle in* G *is its* circumference.

Definition 1.1.18 (Distance) *The* distance $d_G(x, y)$ *in* G *of two vertices* x, y *is the length of a shortest* $x - y$ *path in* G *; if no such path exists, we set* $d(x, y) := \infty$.

Definition 1.1.19 (Diameter) *The greatest distance between any two vertices in* G *is the* diameter *of* G*, denoted by* diamG.

Diameter and girth are, of course, related :

Proposition 1.1.4 *Every graph G containing a cycle satisfies $g(G) \leq 2diamG + 1$.*

Definition 1.1.20 (Connected Graph) *A non-empty graph G is called* connected *if any two vertices are linked by a path in G.*

We finish with 2 concepts that become important in graph partitioning.

Definition 1.1.21 (Matching) *A set M of independent edges in a graph $G = (V, E)$ is called a* matching*. M is a matching of $U \subseteq V$ if every vertex in U is incident with an edge in M. The vertices in U are then called* matched *(by M); vertices not incident with any edge of M are* unmatched*.*

Definition 1.1.22 (k-factor) *A k-regular spanning subgraph is called a k-factor. Thus, a subgraph $H \subseteq G$ is a 1-factor of G if and only if $E(H)$ is a matching of V. If $|V(G)|$ is odd, a 1-factor where one vertex is missing is called* near 1-factor*.*

1.1.3 Computational Complexity and Algorithms

Additional references for this Section can be found in [42, 20].

In computer science, *computational complexity* theory is the branch of the theory of computation that studies the resources, or cost, of the computation required to solve a given computational problem. This cost is usually measured in terms of abstract parameters such as time and space, called computational resources. Time represents the number of steps it takes to solve a problem and space represents the quantity of information storage required or how much memory it takes. There are often tradeoffs between time and space that have to be considered when trying to solve a computational problem. It often turns out that an alternative algorithm will require less time but more space (or vice versa) to solve a given problem. Time requirements sometimes must be amortized to determine the time cost for a well defined average case. Space requirements can be profiled over time, too, especially in consideration of a multi-user computer system.

Complexity theory differs from *computability theory*, which deals with whether a problem can be solved at all, regardless of the resources required.

After the theory explaining which problems can be solved and which cannot be, it was natural to ask about the relative computational difficulty of computable functions. This is the subject matter of computational complexity.

The *time complexity* of a problem is the number of steps that it takes to solve an *instance* (i.e., the problem restricted to a particular value of the input) of the problem as a function of the size of the input (usually measured in bits), using the most efficient algorithm. Of course, the exact number of steps will depend on exactly what machine or language is being used. To avoid that problem, we generally use *Big-O notation.* If a problem has time complexity $O(n^2)$ on one typical computer, then it will also have complexity $O(n^2p(n))$ on most other computers for some polynomial $p(n)$, so this notation allows us to generalize away from the details of a particular computer. Let's formalize this idea.

Definition 1.1.23 (Big-O notation) *If $f, g : \mathbb{R} \to \mathbb{R}$, we say that*

$$f(x) \textit{ is } O(g(x)) \textit{ as } x \to \infty$$

if and only if

$$\exists x_0, \exists M > 0 \textit{ such that } |f(x)| \leq M|g(x)| \ \forall x > x_0$$

Let's talk about another very used notation.

Definition 1.1.24 (Little-o notation) *If $f, g : \mathbb{R} \to \mathbb{R}$, we say that*

$$f(x) \textit{ is } o(g(x)) \textit{ as } x \to \infty$$

if and only if

$$\forall c > 0 \ \exists k > 0 \textit{ such that } 0 \leq |f(x)| < c|g(x)| \ \forall \ x \geq k$$

The value of k must not depend on x, but may depend on c.

Much of complexity theory deals with *decision problems.* A decision problem is a problem where the answer is always YES/NO. A decision problem is equivalent to a *language*, which is a set of finite-length strings. For a given decision problem, the equivalent language is the set of all strings for which the answer is YES. Decision problems are often considered because an arbitrary problem can always be reduced to a decision problem.

An important result in complexity theory is the fact that no matter how hard a problem can get (i.e. how much time and space resources it requires), there will always be even harder problems. For time complexity, this is determined by the time hierarchy theorem. A similar space hierarchy theorem can also be derived.

Complexity theory analyzes the difficulty of computational problems in terms of many different *computational resources.* The same problem can be described in terms of the necessary amounts of many different computational resources, including time, space, randomness, alternation, and other less-intuitive measures. A complexity class is the set of all of the computational problems which can be solved using a certain amount of a certain computational resource.

Perhaps the most well-studied computational resources are *deterministic time* (DTIME) and *deterministic space* (DSPACE). These resources represent the amount of computation time and memory space needed on a deterministic computer, like the computers that actually exist. These resources are of great practical interest, and are well-studied.

Some computational problems are easier to analyze in terms of more unusual resources. For example, a *nondeterministic Turing machine* is a computational model that is allowed to branch out to check many different possibilities at once. The nondeterministic Turing machine has very little to do with how we physically want to compute algorithms, but its branching exactly captures many of the mathematical models we want to analyze, so that nondeterministic time is a very important resource in analyzing computational problems.

1.1.3.1 Complexity classes

Additional references for this Section can be found in [42].

A *complexity class* is the set of all of the computational problems which can be solved using a certain amount of a certain computational resource. The complexity class **P** is the set of decision problems that can be solved by a deterministic machine in polynomial time. This class corresponds to an intuitive idea of the problems which can be effectively solved in the worst cases.

The complexity class **NP** is the set of decision problems that can be solved by a non-deterministic machine in polynomial time. This class contains many problems that people would like to be able to solve effectively, including the *Boolean satisfiability problem*, the *Hamiltonian path problem* and the *Vertex cover problem*. All the problems in this class have the property that their solutions can be checked effectively.

The question of whether P is the same set as NP is the most important open question in theoretical computer science. There is even a $1,000,000 prize for solving it!!!

Questions like this motivate the concepts of hard and complete. A set of problems X is *hard* for a set of problems Y if every problem in Y can be *transformed* easily into some problem in X with the same answer. The definition of "easily" is different in different contexts (the most usual is considering a polynomial transformation). The most important hard set is **NP-hard**. Set X is *complete* for Y if it is hard for Y, and is also a subset of Y. The most important complete set is **NP-complete**.

Problems that are solvable in theory, but can't be solved in practice, are called *intractable*. What can be solved "in practice" is open to debate, but in general only problems that have polynomial-time solutions are solvable for more than the smallest inputs. Problems that are known to be intractable include those that are **EXPTIME-complete**. If NP is not the same as P, then the NP-complete problems are also intractable.
We can see a little diagram of the most important complexity classes in Figure 1.19, provided that $P \neq NP$. If P = NP, then all three classes are equal.

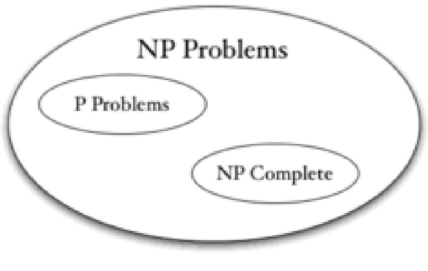


FIG. 1.19 – Diagram of complexity classes provided that $P \neq NP$.

The traditional lines of attack for the NP-hard problems are the following :

- Devising algorithms for finding exact solutions (they will work reasonably fast only for relatively small problem sizes).
- Devising ”suboptimal” or heuristic algorithms, i.e., algorithms that deliver either seemingly or probably good solutions, but which could not be proved to be optimal.
- Finding special cases for the problem (”subproblems”) for which either exact or better heuristics are possible.

1.1.3.2 Parameterized Complexity

In computer science, *parameterized complexity* is a measure of complexity of problems with multiple inputs. It is based on the fact that several such NP-hard are tractable when one of their input is fixed.

The existence of efficient, exact, and deterministic solving algorithms for NP-complete problems problems is considered unlikely, if inputs are not fixed ; all known solving algorithms for these problems require time that is exponential in the total size of the inputs. However, some problems can be solved by algorithms that are exponential in the size of one input and polynomial in the size of the other inputs. Such an algorithm is called a *fixed-parameter algorithm*, because the problem can be solved efficiently by fixing the ”troublesome” input at any one value. A problem that allows for such an algorithm is called *fixed-parameter tractable* (*FPT* for short). Let’s define it a bit more formally.

Definition 1.1.25 *Given a finite alphabet* $\sum^*$*, a* parameterization *is a mapping* $\kappa : \sum^* \rightarrow \mathbb{N}$ *that can be computed in polynomial time. A* parameterized problem *(with respect to* $\sum$*) is a couple* (L, κ) *where* $\mathrm{L} \subseteq \sum^*$ *and* κ *is a parameterization of* $\sum^*$*.*

It is important to remark that there are many possible parameterizations for each problem, and that the complexity of the corresponding parameterized problems can differ very much.

Given an alphabet $\sum$ and a parameterization $\kappa : \sum^* \rightarrow \mathbb{N}$,

Definition 1.1.26 *An algorithm* $\mathcal{A}$ *is a* FPT-algorithm *with respect to* κ *if there exists a computable function* $f : \mathbb{N} \rightarrow \mathbb{N}$ *and a polynomial function* $p : \mathbb{N} \rightarrow \mathbb{N}$ *such that for each* $x \in \sum^*$*, the algorithm* $\mathcal{A}$ *needs* $\leq f(\kappa(x)) \cdot p(|x|)$ *steps.*

Definition 1.1.27 *A parameterized problem* (L, κ) *is* fixed-parameter tractable *is there exists a FPT-algorithm with respect to* κ *that decides* L*. In this case, we say that* $(\mathrm{L}, \kappa) \in FPT$*.*

For example, there is an algorithm which solves the vertex cover problem in $O(kn+1.29^k)$ time, where n is the number of vertices and k is the size of the vertex cover. This proves that vertex cover is fixed-parameter tractable with respect to this parameter, whereas it is well known that this problem is NP-complete.

Definition 1.1.28 *A parameterized problem* (L, κ) *is in* XP *if there exists a computable function* f *and an algorithm such that, given* $x \in \sum^*$*, decides if* $x \in \mathrm{L}$ *in* $O(|x|^{f(\kappa(x))})$ *steps. In this case, we say that* $(\mathrm{L}, \kappa) \in XP$*.*

Proposition 1.1.5 $FPT \subsetneq XP$

1.1.4 Combinatorial Optimization

Combinatorial optimization is a branch of optimization in applied mathematics and computer science, related to operations research, algorithm theory and computational complexity theory that sits at the intersection of several fields, including artificial intelligence, mathematics and software engineering. Combinatorial optimization algorithms solve instances of problems that are believed to be hard in general, by exploring the usually-large solution space of these instances. Combinatorial optimization algorithms achieve this by reducing the effective size of the space, and by exploring the space efficiently.

A study of computational complexity theory helps to motivate combinatorial optimization. Combinatorial optimization algorithms are typically concerned with problems that are NP-hard. Such problems are not believed to be efficiently solvable in general. However, the various approximations of complexity theory suggest that some instances (e.g. ”small” instances) of these problems could be efficiently solved. This is indeed the case, and such instances often have important practical ramifications.

The domain of combinatorial optimization is optimization problems where the set of *feasible solutions* (i.e., the domain of the *objective* or *cost function*) is discrete or can be reduced to a discrete one, and the goal is to find the best possible solution.
An *instance* of a combinatorial optimization problem can be described in a formal way as a tuple $(X, P, Y, f, extr)$ where

- X is the solution space (on which f and P are defined)
- P is the feasibility predicate (a boolean-valued function that determines a subspace of X where an optimal solution must be found)
- Y is the set of feasible solutions (the elements of X for which P gives 1)
- f is the objective function
- $extr$ is the extreme (usually min or max)

1.1.4.1 Heuristics and approximations

Additional references for this Section can be found in [20].

Two fundamental goals in computer science are finding algorithms with provably good run times and with provably good or optimal solution quality. A *heuristic* is an algorithm that gives up one or both of these goals ; for example, it usually finds pretty good solutions, but there is no proof the solutions could not get arbitrarily bad ; or it usually runs reasonably quickly, but there is no argument that this will always be the case.

Often, one can find specially crafted problem instances where the heuristic will in fact produce very bad results or run very slowly ; however, these instances might never occur in practice because of their special structure. Therefore, the use of heuristics is very common in real world implementations. The drawback with heuristic algorithms is that it is difficult to compare them.

Let's introduce now some definitions that will give us the idea of approximations algorithms.

Definition 1.1.29 *The* performance guarantee *of a heuristic algorithm for a minimization (maximization) optimization problem is α if the algorithm is guaranteed to deliver a solution whose value is at most (at least) α times the optimal value.*

Definition 1.1.30 *An $\alpha-$approximation algorithm is a polynomial time algorithm with a performance guarantee of α.*

Typically, if we are in front of a combinatorial optimization problem, we do not know the value of the optimal solution (in that case there would be no such a "problem" !), and what we do is compare the solution of the algorithm with a lower bound (resp. upper bound) of our minimization (resp. maximization) problem. Finding a tight lower/upper bound is sometimes one of the most difficult steps in solving a problem.

1.1.5 Linear Programming

In mathematics, *linear programming (LP)* problems are optimization problems in which the objective function and the constraints are all linear.

Linear programming is an important field of optimization for several reasons. Many practical problems in operations research can be expressed as linear programming problems. Certain special cases of linear programming, such as network flow problems and multicommodity flow problems are considered important enough to have generated much research on specialized algorithms for their solution. A number of algorithms for other types of optimization problems work by solving LP problems as sub-problems. Historically, ideas from linear programming have inspired many of the central concepts of optimization theory, such as duality, decomposition, and the importance of convexity and its generalizations.

Standard form is the usual and most intuitive form of describing a linear programming problem. It consists of the following three parts :

- A *linear function to be maximized.* For instance :

$$\text{Maximize } c_1x_1 + c_2x_2$$

- *Problem constraints* in a standard form. For instance :

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 \leq b_2$$
$$a_{31}x_1 + a_{32}x_2 \leq b_3$$

- *Non-negative variables.* For instance :

$$x_1 \geq 0$$
$$x_2 \geq 0$$

It is usual to express the problem in *matrix form*, and then it becomes :

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Every linear programming problem, referred to as a *primal problem*, can be converted into an equivalent *dual problem*. In matrix form, we can express the *primal problem* as :

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

The equivalent *dual problem* is :

$$\begin{aligned} \text{minimize} \quad & b^T y \\ \text{subject to} \quad & A^T y \geq c \\ & y \geq 0 \end{aligned}$$

where y is used instead of x as variable vector.

Geometrically, the linear constraints define a *convex polyhedron*, which is called the *feasible region*. We can see an example of feasible region in Figure 1.20. Since the objective function is also linear, all local optima are automatically global optima. The linear objective function also implies that an optimal solution can only occur at a boundary point of the feasible region.

The **simplex algorithm**, developed by George Dantzig, solves LP problems by constructing an admissible solution at a vertex of the polyhedron, and then walking along edges of the polyhedron to vertices with successively higher values of the objective function until the optimum is reached. Although this algorithm is quite efficient in practice (in fact, simplex is the most used algorithm in the world, even more than the FFT, which is the second), and can be guaranteed to find the global optimum if certain precautions against *cycling* are taken, it has poor worst-case behavior : it is possible to construct a linear programming problem for which the simplex method takes a number of steps exponential in the problem size. In fact for some time it was not known whether the linear programming problem was solvable in polynomial time (complexity class P). The first worst-case polynomial-time algorithm for the linear programming problem was proposed by Leonid Khachiyan in 1979.

If the unknown variables are all required to be integers, then the problem is called an **integer programming (IP)** or **integer linear programming (ILP)** problem. In contrast to linear programming, which can be solved efficiently in the worst case, integer programming problems are in the worst case undecidable, and in many practical situations (those with bounded variables) NP-hard. 0-1 integer programming is the special

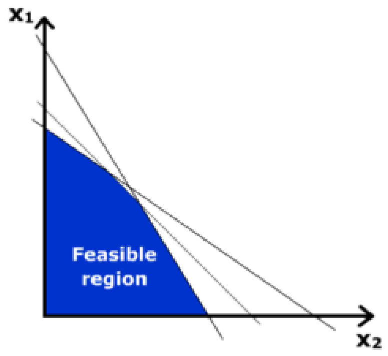


Fig. 1.20 – A series of linear constraints on two variables produces a region of possible values for those variables

case of integer programming where variables are required to be 0 or 1 (rather than arbitrary integers). This method is also classified as NP-hard.

If only some of the unknown variables are required to be integers, then the problem is called a **mixed integer programming (MIP** or **MILP)** problem. These are generally also NP-hard.

There are however some important subclasses of IP and MIP problems that are efficiently solvable, most notably problems where the constraint matrix is *totally unimodular* and the right-hand sides of the constraints are integer.

1.1.6 Design Theory : combinatorial designs

Additional references for this Section can be found in [15, 39, 45].

Roughly, *design theory* consists in packing the elements of a given set into blocks under certain constraints. A particular packing is called a *combinatorial design* or, more often, simply *design*. This area of discrete mathematics has application to many fields, and it has itself a wide amount of new concepts and important results. Here we will restrict to the definition of the concepts that we will use in our work. Maybe the most used handbook in combinatorial designs is [15], by Charles J. Colbourn and Jeffrey H. Dinitz.

Definition 1.1.31 (t-(v, k, λ) design) *A* t-(v, k, λ) design *is a pair* $(X, \mathcal{B})$ *where* X *is a* v*-element set of* points *and* $\mathcal{B}$ *is a collection of* k*-element subsets of* X *(*blocks*) with the property that every* t*-element subset of* X *is contained in exactly* λ *blocks.*
A design is simple *if no two blocks are identical.*
A t-(v, k, λ) design is also denoted by $S_\lambda(t, k, v)$*. If* $\lambda = 1$*, then it can be omitted.*

The following definition is a particular case of the previous one, but that has his own name because of his great importance in practice. In fact, this is the natural structure that appears when we have sets of nodes that want to communicate among them (under the assumption that each communication takes place between a pair of nodes), as it is the case in our work.

Definition 1.1.32 (BIBD) *A 2-(v, k, λ) design is called a* balanced incomplete block design *(*BIBD *for short). In this case we can extend a bit the usual definition : A* balanced incomplete block design *(*BIBD*) is is a pair* $(V, \mathcal{B})$ *where* V *is a* v*-set and* $\mathcal{B}$ *is a collection of* b k*-subsets of* V *(*blocks*) such that each element of* V *is contained in exactly* r *blocks and any* 2*-subset of* V *is contained in exactly* λ *blocks. The numbers* v, b, r, k, λ *are* parameters *of the* BIBD*.*

Proposition 1.1.6 *Trivial necessary conditions for the existence of a* BIBD(v, b, r, k, λ) *are*
(1) $vr = kb$
(2) $r(k-1) = \lambda(v-1)$
Parameter sets that satisfy (1) and (2) are called admissible.

Other widely used construction is the *Steiner triple system*, which is also a particular case of the general definition.

Definition 1.1.33 (Steiner system) *Given three integers* t, k, v *such that* $2 \leq t < k < v$*, a* Steiner system S(t, k, v) *is a* v*-set together with a family* $\mathcal{B}$ *of* k*-subsets of* V *(*blocks*) with the property that every* t*-subset of* V *is contained in exactly one block. Equivalently, an* S(t, k, v) *is a t-(v, k, 1) design.*

Definition 1.1.34 (STS) *An S(2, 3, v) is a* Steiner triple system STS*(v). Equivalently, an STS(v) is a 2-(v, 3, 1) design.*

The last ingredient of design theory that we will need is the concept of covering.

Definition 1.1.35 (Covering) *A* t-(v, k, λ) covering *is a pair* $(X, \mathcal{B})$ *where* X *is a* v*-set of* points *and* $\mathcal{B}$ *is a collection of* k*-subsets of* X *(*blocks*) with the property that every* t*-subset of points occurs in at least* λ *blocks in* $\mathcal{B}$*. Repeated blocks in* $\mathcal{B}$ *are permitted.*

Definition 1.1.36 (Covering number) *The* covering number C_λ(v, k, t) *is the minimum number of blocks in any t-(v, k, λ) covering.*
A t-(v, k, λ) covering $(X, \mathcal{B})$ *is* optimal *if* $|\mathcal{B}| = C_\lambda(v, k, t)$.
If $\lambda = 1$*, then write C(v, k, t) for* C_1*(v, k, t).*

1.2 The Traffic Grooming Problem

1.2.1 Introduction

Traffic grooming in WDM networks is defined as the allocation of sub-wavelength traffic tributaries onto full wavelengths channels in order to achieve efficient utilization of network resources, such that some cost function be minimized. This is usually implemented according to a strategy that optimizes a certain objective function, such as minimizing cost or blocking probability, or maximizing revenue. Realization that most of the application bandwidth requirements are sub-wavelength has put the traffic grooming under the spotlight, and increased its importance. In the beginning, the motivation for traffic grooming was merely reducing the total number of required wavelengths, such that, given limited number of wavelengths, maximum amount of traffic can be accommodated. However, the understanding that traffic grooming can significantly reduce the number of higher layer components, and thus network cost, has created an enormous interest in this area.

ADMs

A key component in the SONET network is the add-drop multiplexer ADM, which handles the pass-through traffic, demultiplexes the traffic to destinations (drop), and adds traffics to the network. The optical add-drop multiplexer (OADM) which works at the optical domain can handle the pass-through traffics, drop traffics, and add traffics in optical channels. See Section 1.1.1 for a detailed explanation of these components. For each optical channel dropped (or added), an SONET ADM is needed to convert the optical signal to electronic ones (electronic signals to optical ones). If a dropped channel and an added channel at a node are given the same wavelength, the two channels can share the same ADM at the node. If a channel only optically pass-through a node, there is no necessary to have an ADM for that channel at the node. Since ADMs are expensive and major components of the SONET network, minimizing the number of ADMs is an important research area and the optimization problem is known as traffic grooming. To realize a given set of traffic demands using the minimum number of ADMs, we should route the traffics to produce as many pass-through channels as possible.

Routing with protection

Due to the high data rate in the optical networks, a failure may results in a huge loss of data. To overcome this problem, routing with protection (using additional fibers/-channels as back-up) is common in the SONET rings. There are two protection switching technologies. One is the unidirectional protection switching in which the traffic of each direction is handled independently. If there is a failure on a single fiber, only the traffic on that fiber (one direction) is switched to the protection fiber and there is no change on the other direction. The other is the bidirectional protection switching in which the traffics on both directions are handled together. This means that if there is a failure on one direction of transmission (one fiber) then the traffics on both directions are switched to the protection fibers. Based on the protection switching technologies, there are two common types of SONET rings in practice : the unidirectional path-switched rings (*UPSR*) and the bidirectional line-switched rings (*BLSR*). In addition to the routing and wavelength assignment problems, the routing algorithms for the SONET ring also need to deal with the protection problem. This makes the algorithms special.

Traffic grooming on SONET rings

SONET rings with point-to-point wavelength-division multiplexing (WDM) links are widely used in practice. In such SONET/WDM networks, each optical fiber can support a number of wavelength channels, and each wavelength channel may have a bandwidth of a few gigabit per second. In order to make use of the wavelength capacity efficiently, it is critical to groom the low-rate traffic streams (i.e., multiplex multiple low-rate traffic streams into a high-speed wavelength channel). The ratio of the wavelength channel rate to the lowest traffic stream rate is known as the grooming factor. For example, sixteen OC-3 low-rate traffic streams can be multiplexed into an OC-48 wavelength channel, in which case the grooming factor is 16. We can see an example of what does the grooming factor mean in Figure 1.21.

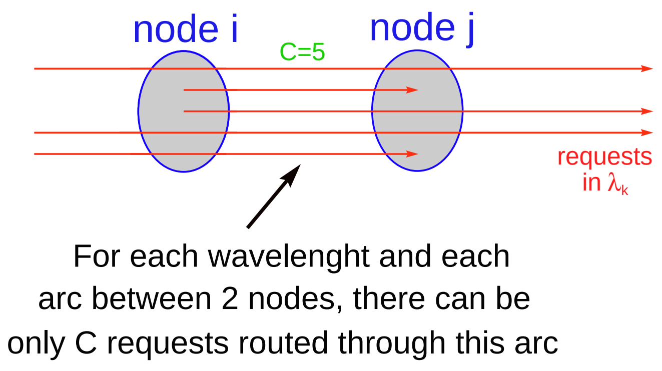

node i
node j
C=5
requests in λ_k
For each wavelenght and each arc between 2 nodes, there can be only C requests routed through this arc

FIG. 1.21 – Illustrating the definition of *grooming factor*

Traffic grooming is realized by SONET ADMs, which dominate the cost of a SONET/WDM network. With the optical (or wavelength) add-drop multiplexer (OADM), it is possible for a node to optically bypass the wavelength channels that do not carry low-rate traffic streams from/to the node. Therefore, due to the deployment of OADMs, ADMs are needed in each node only for the wavelength channels that carry low rate traffic streams from/to the node. The traffic grooming problem is to find a grooming scheme for a given traffic pattern with the presence of one OADM at each node, such that the number of required ADMs is minimized. We will state formally this problem in the next Section.

1.2.2 Statement of the Problem

As we have already said, *Traffic grooming* in networks refers to group low rate traffic into higher speed streams (see [25, 40, 44, 35, 27] for related surveys). There are many variants according to the type of network considered (for example, in [4, 9] the Path grooming problem is studied), the constraints used and the parameters one wants to optimize which give rise to a lot of interesting design problems (graph decomposition).

To fix ideas, suppose that we have an *optical network* represented by a directed graph G (in many cases a symmetric one) on N vertices, for example an unidirectional ring $\vec{C}_N$ or a bidirectional ring C_N^*. We are given also a traffic (or instance) matrix, that is a family of connection requests represented by a multidigraph I (the number of arcs from i to j corresponding to the number of requests from i to j). An interesting case is when there is exactly one request from i to j ; then $I = K_n^*$. It is usual to refer to G as the *physical graph*, whereas I is referred as the *logical graph* (or *request graph*). Satisfying a request r from i to j consists in finding a route (path) $P(r)$ in G and assigning it a wavelength ω.

The objective is to minimize the equipment cost. Among possible criteria, one is to minimize the number of wavelengths used to route all the requests. This leads to the widely studied *loading problem* [3, 24]. Another choice, which is in fact a better approximation of the true equipment cost, is to minimize the number of add/drop locations (namely ADMs using SONET terminology) instead of the number of wavelengths. This leads to the *grooming problem*, that we will state formally later. These two problems are proved to be different. Indeed, it is known that even for the simpler network (the unidirectional ring), the number of wavelengths and the number of ADMs cannot be simultaneously minimized [13, 29].

The *grooming factor* (or *grooming ratio*) C (or g depending on the article) means that a request uses only $\frac{1}{C}$ of the bandwith available on a wavelength along its route. Said otherwise, for each arc e of G and for each wavelength w, there are at most C paths of wavelength w containing arc e. For each wavelength, an ADM is needed at each node used in the ends of a lightpath followed by a request. We can formalize this idea by introducing the load definition :

Definition 1.2.1 *For a subgraph B_ω of requests of I, we define the* load *of an edge e of G, $L(B_\omega, e)$, as the number of requests which are routed through e. I.e :*

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

Minimize the number of ADMs corresponds to minimize the total number of vertices of all the subgraphs B_ω of I. The solutions obtained might differ if we impose restrictions on the routing. In order to formalize this idea, we introduce the following definition.

Definition 1.2.2 *For each request r, note by A_r the set of allowed paths in G for this request, and let $A = \bigcup_{r \in I} A_r$.*

We can see in Table 1.2 a summary of the translation of the parameters of the Problem from Telecommunications to Mathematics.

Optical Network parameters	**Graph Model**
Optical Network topology	Digraph G
Traffic Matrix, with instances of requests (i, j)	Digraph I, with arcs (i, j)
Routing a request r	Assigning a dipath $P(r)$ in G and a wavelength ω
Requests on wavelength ω	Subgraph B_ω of I
Grooming Factor C (a request uses $\frac{1}{C}$ of the available bandwidth on a wavelength ω)	$\forall$ arc $e \in G$ and $\forall \omega, \exists$ at most C paths on wavelength ω containing arc e : $L(B_\omega, e) \leq C$
ADM on wavelength ω	Vertex in subgraph B_ω
Restrictions on the routing	Set A of allowed paths
Objective : minimize total number of ADMs	Objective : minimize $\sum_{\omega=1}^{W} \|V(B_\omega)\|$

Tab. 1.2 – Translation of the Problem from Telecommunications to Mathematics

At this moment we are already able to state the general Traffic Grooming Problem.

Problem 1.2.1 (The Traffic Grooming Problem)
Input : *A digraph G (network), a digraph I (set of requests), a set A (allowed paths) and a grooming factor C*
Output : *Find for each arc $r \in I$ a path $P(r) \in A_r$, and a partition of the arcs of I into subgraphs B_w, $1 \leq w \leq W$, such that $\forall e \in E(G)$ $L(B_\omega, e) \leq C$*
Objective : *Minimize $\sum_{w=1}^{W} |V(B_w)|$, and this minimum is denoted $A(G, I, A, C)$*

We will see examples when we focus on a particular case in Chapter 2.

1.2.3 Relation with the Loading Problem

Our aim in the definition of the grooming problem was to minimize the total number of ADMs. Another possibility is to minimize the number of wavelengths needed in the decomposition of the request graph. This leads to the loading problem, widely studied in the literature ([28, 21, 36, 31, 41, 43]). The ring-grooming problem has the advantage of being easy to state and amenable to rigorous analysis, but has the drawback that its cost function gives only an approximation to the true cost of building a SONET ring network. In some situations, like SONET over WDM which is the case under study here, a better approximation to the cost may be proportional to the number of its *add/drop points*, that is, ADMs. Hence, the grooming problem represents a better approximation to the true cost of SONET networks, but on the other hand it becomes more difficult to solve than the loading problem.

In view of the similarity between the loading and the grooming problem, we might wonder whether both problems are equivalent.

In [13, 36] the authors prove that even in the simplest network (the unidirectional ring) the number of ADMs and the number of wavelengths cannot be simultaneously minimized. We should give an example of this. Let $N = 9$ and $C = 1$, and consider the scenario of a unidirectional ring. Let the list of traffic demands be

$$\{1, 2\}, \{3, 1\}, \{2, 3\}, \{4, 5\}, \{6, 4\}, \{5, 6\}, \{7, 8\}, \{9, 7\}, \{8, 9\}$$

We can see an illustration in Figure 1.22, where the blue requests are routed via the longest path, because we are in a unidirectional ring.

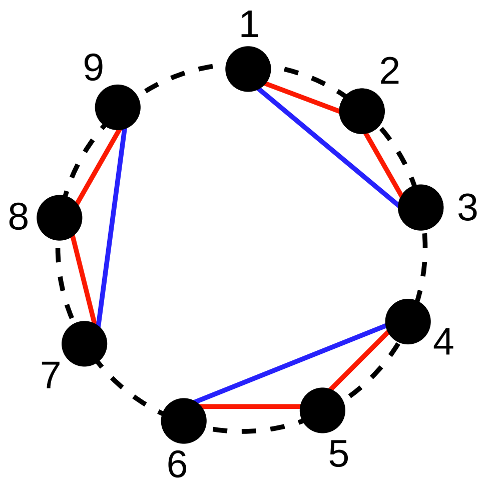


FIG. 1.22 – Example about the relation with the *loading problem*

If we want to minimize the number of ADMs, since we have requests in all the nodes, the best we can do is to place a single ADM at each node. Since each of the 3 traffic triangles covers the whole load, we must place them in different wavelengths. Thus, the best grooming solution uses three rings (i.e. 3 wavelengths) : one with ADMs at vertices $\{1, 2, 3\}$, one with ADMs at vertices $\{4, 5, 6\}$ and another one with ADMs at vertices $\{7, 8, 9\}$. In this way, we use 3 wavelengths and 9 ADMs.

On the other hand, if we want to minimize the number of wavelengths, realize that we cannot do it with only 1 ring, because there are arcs with load 2. Thus, the best we can do is to use 2 wavelengths, and this can be done for example with a ring with ADMs at all the vertices, and another one with ADMs at vertices $\{1, 3, 4, 6, 7, 9\}$. From this example we can conclude that, in general, the number of wavelengths and the number of ADMs cannot be simultaneously minimized.

We can prove that the problems are not equivalent in a more general and simple way. We need the notion of *traffic splitting*, that means that the traffic demand between a given pair of nodes can be split in both directions of the ring. The ring loading allowing traffic splitting can be solved in polynomial time [28]. On the other hand, ring grooming with traffic splitting is NP-complete (Section 1.2.5). Thus, provided that $P \neq NP$ (maybe the most accepted conjecture in the world!) and because of the collapse of the polynomial hierarchy (is there exists one problem that belongs to P and NP-complete, then $P = NP$), we conclude that both problems must belong to different complexity classes, and therefore they are not equivalent.

1.2.4 State-of-the-art

A huge amount of research as been done in this area in the last few years. Here we will give a brief summary about the current state of the research in this topic, we do not pretend at all to build an exhaustive survey.

Traffic grooming appears in the context of optical networks. As it is logic, many of the researches that work in this area have above all a technical background, and thus most of the tools that have been applied to the problem of traffic grooming are MILP formulation, heuristics and approximation algorithms. Of course these are extremely useful and efficient tools, but we are interested in developing more theoretical approaches to the problem.

Let's focus mainly on the case of the grooming problem in the Ring, which is a really used topology in optical networks, above all in backbone networks. Unidirectional and bidirectional case must be handled independently.

1.2.4.1 Unidirectional Ring

In the **unidirectional ring grooming problem**, *Bermond, Coudert* and *Muñoz* [10] found a translation of the problem that has allowed the publication of further results. Since good results have been found using this method, we will not deal here with the other possible approaches to the problem. The idea of this translation that we have talked about is the following :

In the unidirectional ring, each pair of requests $(i,j),(j,i)$ induces load 1 in all the arcs of the ring, because both requests must be routed through disjoint arcs. We can think of such a pair or requests as an arc between the 2 nodes, as it is shown in Figure 1.23.

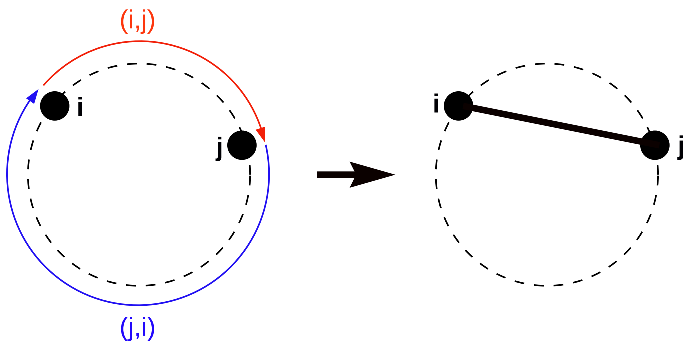


FIG. 1.23 – Simplification of a pair of requests in the unidirectional case

In this way, the problem can be reformulated as decomposing a complete graph into subgraphs with at most C edges (this is the load constraint) with the objective of minimizing the total number of vertices. At this stage, an avid reader could object that with this simplification we are imposing that the requests (i,j) and (j,i) must be routed through the same wavelength. It is a logic remark, because we have not imposed this restriction a priori in the statement of the general grooming problem. Fortunately, *Charles J. Colbourn* has proved (personal communication) that this constraint does not affect the value of the optimal solution.

Nowadays, the problem has been solved for values of the grooming factor until 6 :

- With this reformulation, the cases $C = 1$ and $C = 2$ become trivial, because it it easy to state when a complete graph can be decomposed into subgraphs with 1 or 2 edges.
- In [5] the case $C = 3$ is completely solved (2003).
- In [33, 10] the case $C = 4$ is completely solved (2002, 2003).
- In [7] the case $C = 5$ is completely solved (2004).
- In [8] the case $C = 6$ is completely solved (2005).

- In [10] some particular cases for several values of C are solved using design theory (2003).

But, as C grows, the constructions are more and more complicated, and consequently most of the authors that have worked hard in this problem during the last years have decided to give up the task of finding the optimal solution for greater values of C $(7, 8, \ldots)$.

1.2.4.2 Bidirectional Ring

In the **bidirectional ring grooming problem** the scenario is quite different. There is still an important lack of a theoretical solution of the problem. In contrast to the unidirectional case, there is not a similar simplification, and that's the reason that makes this problem harder than the other one. Nevertheless, its study has attracted the interest of numerous researchers :

- In [34] a MILP formulation of the problem can be found.

- There are a lot of articles providing heuristics about the ring grooming. See for instance [44, 24, 13, 29, 30, 12].

- Up to date, the algorithm with the best approximation ratio for a general instance of requests has been found by *Flammini, Moscardelli, Shalom* and *Zaks* [27] (2006). Their algorithm has ratio $2\log(C) + o(\log(C))$ regardless of the routing used in the ring.

- *Colbourn* and *Wan* [18] (2001) applied tools from design theory to the bidirectional case. Their method is based in the idea of *primitive rings*. Nevertheless, they don't prove lower bounds and they don't care about the optimality of the solutions that they obtain.

- *Chow* and *Lin* [14] (2004) prove a new lower bound for the bidirectional ring (regardless of routing). We will talk about this lower bound later. They consider the case when there is a traffic demand between each pair of nodes. This kind of traffic is called *K-quasi-uniform*, where K is the largest traffic demand divided by the smallest one. They find a constant-factor approximation algorithm with ratio $\max(2K, 12\sqrt{2K})$ for *K-quasi-uniform* traffic, which is the best one up to date.

1.2.4.3 Path

Also the **path grooming problem** has been studied using theoretical approaches, similar to the ones that have been used for the ring. In [4] the cases C=1 and C=2 are solved and in [9] the maximum number or requests that can be groomed on the path (given a grooming factor) is found.

1.2.5 Complexity of the Problem

Like most of the problems of combinatorial optimization, the traffic grooming problem has been proved to be NP-complete. Anyway, there are still some open problems about the complexity of this problem, and this is the topic that we would like to present in this Section.

Let's focus in the case where the topology a a ring. In terms of standard complexity, the ring grooming problem has been proved to be NP-complete, for instance by reducing the well known NP-complete problem of *bin packing* to it :

Proposition 1.2.1 ([14]) *Ring grooming is NP-complete.*

Nevertheless, if the size of the ring is fixed, then the ring grooming problem turns out to be solvable in polynomial time :

Proposition 1.2.2 ([14]) *If N is fixed, then ring grooming is solvable in polynomial time.*

As one can check in the proof of Proposition 1.2.2 the practical problem is that the degree of the polynomial depends on N. In terms of parameterized complexity, this proves that the ring grooming problems belongs to XP, but we still don't know if it belongs to FPT, because as we know from Proposition 1.1.5, $FPT \subsetneq XP$.

If we take the number of ADMs to be the parameter of the problem, then it has been proved (*Michael Fellows, University of Newcastle*) that ring grooming is in FPT, but it is still an open problem if it belongs to FPT if N is the parameter.

1.3 Concluding Remarks

After providing all the necessary concept for understanding this work, we have formulated the Traffic Grooming problem in terms of graph partitioning, and we have realized that it turns out to be a problem of combinatorial optimization.

We have also talked about the previous work that has been done, and finally we have seen that the Grooming Problem is NP-complete.

Chapter 2

Approaches to the All-to-all Bidirectional Ring case

Abstract

In this Chapter we focus on the specific case that we have studied : the Bidirectional Ring Grooming Problem with all-to-all unitary requests and shortest path symmetric routing. We make a mathematical formulation and we find results that allow us to solve the problem for particular cases.

2.1 Statement of the Problem in this case

In Section we have formally stated the general Traffic Grooming Problem. Many versions of the problem can be considered, according for example to the routing and the graphs G and I, among others.
If the network topology is a *ring*, we can mainly distinguish two cases depending on the type of routing that we use in the optical network under analysis :

- Unidirectional case : we can route the requests only following one direction in the cycle. See for instance [22, 8, 10, 7, 5].
- Bidirectional case : we can route the requests either clockwise $\circlearrowright$ or counterclockwise $\circlearrowleft$. This case has been much less studied than the unidirectional one, due to its higher complexity. See for instance [18, 14].

We will focus here on the **Bidirectional Ring Grooming Problem**, and specifically the **all-to-all unitary case**.

Let's see now how the restrictions on the routing can affect the solutions. For instance, consider a **symmetric routing**, that means that if (i, j) is routed in one direction, then (j, i) is routed by the symmetric dipath, i.e. in the other direction. In this situation, let be $N = 4$ and $C = 2$. The only way to obtain $W = 2$ with load 2 is to route each request $(i, i+1)$ (resp. $(i, i-1)$ in the other subgraph) via the arc $(i, i+1)$ (resp. $(i, i-1)$ in the other subgraph), $(0, 2)$ and $(2, 0)$ in the same direction, and $(1, 3)$ and $(3, 1)$ in the opposite direction (i.e. in the other subgraph). But then the paths associated to $(0, 2)$ and $(2, 0)$ are not symmetric.
Another example is obtained by considering $|V(B_\omega)| = 3$, $V(B_\omega) = \{A, B, C\}$ and C=2. If we have symmetric routing we can have at most 3 requests in a given direction, for instance $\vec{AB}, \vec{BC}$ and $\vec{CA}$, or $\vec{AB}, \vec{BC}$ and $\vec{AC}$, but if we admit not symmetric routing then we can combine 4 requests in one direction, for example $\vec{AB}, \vec{BC}, \vec{AC}$ and $\vec{CA}$. In this case the problem can be reformulated as follows :

Problem 2.1.1 (Form 1 : Symmetric routing)
Input : C_N^* *: bidirectional cycle oriented in both directions, and a grooming factor* C*. All to all requests, and the requests* (i, j) *and* (j, i) *being routed through opposite*

directions

Output : *Find for each arc $r \in K_N^*$ a path $P(r)$ in C_N^*, and a partition of the arcs of K_N^* into subgraphs B_w, $1 \leq w \leq W$, such that $\forall e \in E(C_N^*)$ $L(B_\omega, e) \leq C$*

Objective : *Minimize $\sum_{w=1}^{W} |V(B_w)|$*

We can also impose a **shortest path routing**, but that makes some solutions not compatible with the load constraint, like for $C = 1$, and three requests $\overrightarrow{AB}, \overrightarrow{BC}, \overrightarrow{CA}$ with $d(A, B) + d(B, C) < \frac{N}{2}$.

We can distinguish two cases, depending on the symmetry of the longest requests when N is even. Of course, we could also not impose any kind of symmetry, but then the problem becomes slightly different. In the non-symmetric case the problem can be reformulated as follows.

Problem 2.1.2 (Form 2 : Non-Symmetric Shortest Path routing)

Input : *C_N^* : bidirectional cycle oriented in both directions, and a grooming factor C. A set of requests given by K_N^*, and the request (i, j) being routed by the shortest path. The shortest path from i to j follows the direction of the cycle.*

Output : *Find for each arc $r \in K_N^*$ a path $P(r)$ in C_N^*, and a partition of the arcs of K_N^* into subgraphs B_w, $1 \leq w \leq W$, such that $\forall e \in E(C_N^*)$ $L(B_\omega, e) \leq C$*

Objective : *Minimize $\sum_{w=1}^{W} |V(B_w)|$*

It is not difficult to realize that not always neither symmetric routing implies shortest path routing, nor shortest path routing implies symmetric routing.

If we impose both shortest routing and symmetric routing, the problem is equivalent to consider $G = \vec{C}_N$ and $I = T_N$, where T_N is a *tournament* formed by all the arcs $(i, i+q)$, $i = 0, \ldots, N-1$, $q = 1, \ldots, \lfloor \frac{N}{2} \rfloor$ (plus $\frac{N}{2}$ arcs of the form $(i, i + \frac{N}{2})$, if N is even. Pairs of the form $(i, i + \frac{N}{2})$, $(i + \frac{N}{2}, i)$ are not allowed).

We will focus on the **Bidirectional Ring Grooming Problem** with **symmetric shortest path routing**, and specifically the **all-to-all unitary case**.

Remark 2.1.1 *In this case, we minimize the number of ADMs used by the requests following one direction in the cycle, and then double the number of ADMs and the number of wavelengths to compute the total number of ADMs used by the whole set of requests. However, all the results that we will show take into account only half of the total number of ADMs. Unless necessary, we will not remind that the total needed number of ADMs is twice the number of ADMs of our results.*
In this way, we can get rid of the orientation of the requests, because all of them have the same direction. This is the main reason for choosing this routing, besides of its common use in real optical networks.

I.e., from now on :

$$G = \vec{C}_N \quad and \quad I = T_N.$$

Finally, our problem can be reformulated as follows.

Problem 2.1.3 (Form 3 : Symmetric Shortest Path routing)

Input : $\vec{C}_N$ *: unidirectional cycle, and a grooming factor* C*. A set of requests given by a tournament* T_N *formed by all the arcs* $(i, i+q)$*,* $i = 0, \ldots, N-1$*,* $q = 1, \ldots, \lfloor \frac{N}{2} \rfloor$ *(plus* $\frac{N}{2}$ *arcs of the form* $(i, i+\frac{N}{2})$*, if* N *is even. Pairs of the form* $(i, i+\frac{N}{2})$*,* $(i+\frac{N}{2}, i)$ *are not allowed). The request* (i,j) *being routed by the shortest path, and the requests* (i,j) *and* (j,i) *being routed through opposite directions*

Output : *Find for each arc* $r \in T_N$ *a path* $P(r)$ *in* $\vec{C}_N$*, and a partition of the arcs of* K_N *into subgraphs* B_w*,* $1 \leq w \leq W$*, such that* $\forall e \in E(\vec{C}_N)$ $L(B_\omega, e) \leq C$

Objective : *Minimize* $\sum_{w=1}^{W} |V(B_w)|$

Thus, in the whole work we will deal with the formulation of Problem 2.1.3.

Definition 2.1.1 *Given a bidirectional ring on* N *nodes with symmetric shortest path routing and a grooming factor* C*, the optimal solution of the Traffic Grooming problem is noted by* $A(C, N)$*.*

In the following remarks we can see that Problem 2.1.2 and Problem 2.1.3 are not always equivalent.

Remark 2.1.2 *If N is odd, Problem 2.1.2 and Problem 2.1.3 are equivalent.*

Proof: Let $N = 2q + 1$. The only requests that may be routed differently according to the two Problems are the "diameters" in the even case. Exactly, if the length of the request is smaller than $\lceil \frac{N}{2} \rceil$, then the shortest path is unique and thus Problem 2.1.3 and Problem 2.1.2 give the same routing. In the odd case, the longest request has length q, which is smaller than $\lceil \frac{N}{2} \rceil = q + 1$. □

Remark 2.1.3 *If N is even, Problem 2.1.2 and Problem 2.1.3 are not equivalent.*

Proof: Let $N = 2q$. The difference resides in the fact that the routing is different for the requests of length q. Following the formulation of Problem 2.1.3, the requests $(i, i+q)$ and $(i+q, i)$ are routed via the same edges of the cycle, in opposite directions. On the other hand, following the formulation of Problem 2.1.2, the requests $(i, i+q)$ and $(i+q, i)$ are routed via disjoint edges of the cycle, both in the same direction.
For instance, let $N = 4$ and $C = 2$. Let the vertices be ABCD.
Using Form 1, the best we can do is to use one graph on 4 vertices with the requests AB BC CD DA AC, and another one only with the request BD. Using the same graphs inverting the directions of the requests, we obtain 12 ADMs.
Using Form 2, the best we can do is to use one graph on 4 vertices with the requests AB BC CD DA AC CA, and another one in the opposite direction with the requests BA AD DC CB BD DB, obtaining in this way 8 ADMs. □

It is time now to give some examples.

Example 2.1.1 *$C = 1$, $N = 5$ (10 requests).*
The 3 graphs :
013 with arc 01 13 and 30 ;
124 with arcs 12, 24 and 41 ;
0234 with arcs 02 23 34 and 40
form optimal solution with 10 ADMs

Example 2.1.2 *N=13 C= 3 an optimal solution with 39 ADMs is obtained with :*
The three K_5 : 0 1 4 7 10 ; 0 2 5 8 11 ; 0 3 6 9 12 ;
and the 4 $K_6 - 3e$: 1 2 3 7 8 9 ; 1 5 6 7 11 12 ; 2 4 6 8 10 12 ; 3 4 5 9 10 11.
In a K_5 ABCDE we take the arcs AB BC CD DE EA AC BD CE DA EB, and in a $K_6 - 3e$ ABCDEF we take the arcs AB BC CD DE EF FA AC BD CE DF EA FB.
One can check that all the 78 requests appear exactly once.

In Figure 2.1 we decompose T_7 using 2 wavelengths and 14 ADMs when $C = 3$.

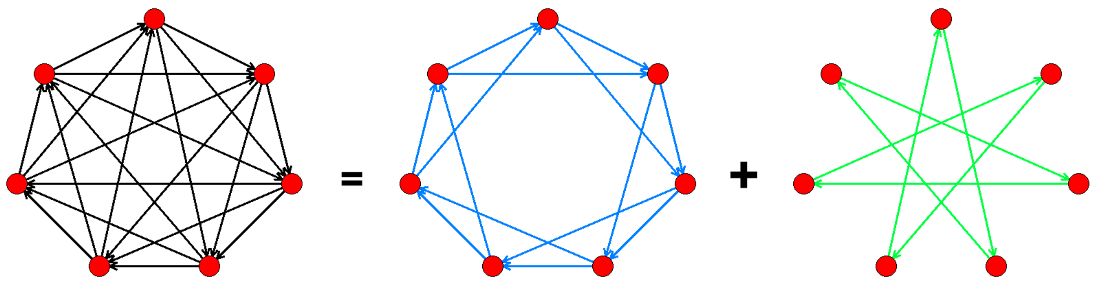


FIG. 2.1 – Example of a decomposition with grooming factor 3

In Figure 2.2 we decompose T_5 in two different ways when $C = 2$, using in both decompositions 2 wavelengths, but different number of ADMs.

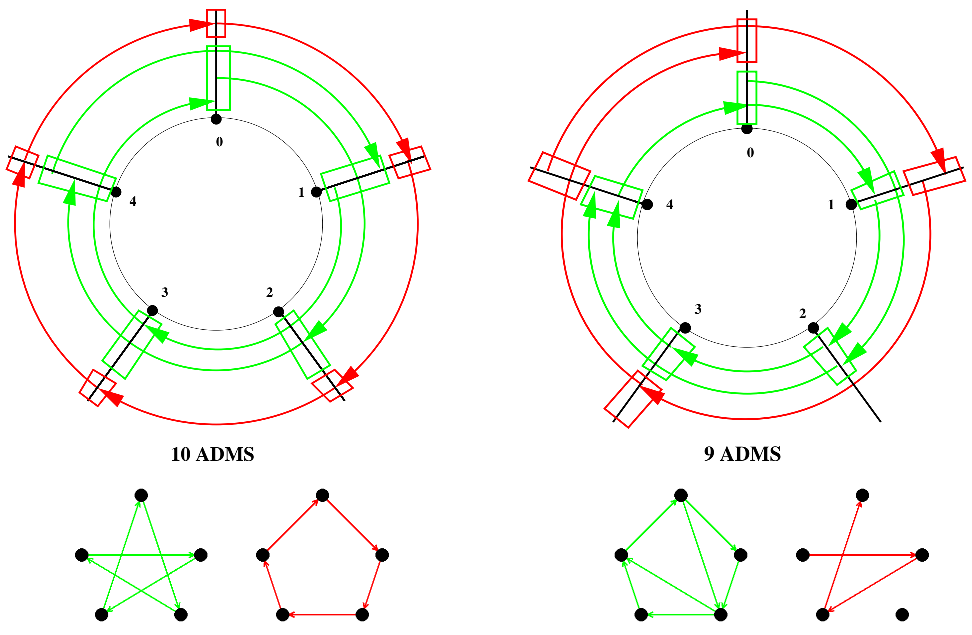


FIG. 2.2 – Decomposing T_5 in two different ways

2.1.1 Mathematical formulation

Let's introduce some notation :

Consider a valid construction for the Problem and let a_p denote the number of subgraphs of the partition with exactly p nodes, A the number of ADMs, and W the number of subgraphs of the partition.

We have the following equalities :

$$A = \sum_{p=2}^{N} p a_p \tag{2.1}$$

$$\sum_{p=2}^{N} a_p = W \tag{2.2}$$

$$\sum_{w=1}^{W} |E_w| = |E| \tag{2.3}$$

In the particular case where $I = T_N$, we have $|E| = \frac{N(N-1)}{2}$, and we know that,

$$W \geq \left\lceil \frac{N^2 - \varepsilon}{8C} \right\rceil, \text{ where } \varepsilon = \begin{cases} 0, & \text{if } N \text{ even} \\ 1, & \text{if } N \text{ odd} \end{cases}$$

Let's recall the proof :

If $N = 2q+1$, then the load of each arc is $1+2+\ldots+q = \frac{N^2-1}{8}$. Dividing these value by C we obtain the minimum number of subgraphs.

If $N = 2q$, then the total load in all the arcs is $N+2N+3N+\ldots+(q-1)N+q\frac{N}{2} = \frac{N^3}{8}$. Since there are N arcs, dividing this value by N and by C we obtain that we need at least $\frac{N^2}{8C}$ subgraphs.

2.1.2 About the length of the requests

Before defining and finding the value of $\gamma(C,p)$ (in Proposition 2.2.1) it is convenient to clarify what we mean by the *length* of a request. Note that the subgraphs for which we will define $\gamma(C,p)$ are inside the ring. But, unless we say it explicitly, the distance we will consider is the distance *inside* the subgraph, not the actual distance in the ring. Anyway, let's formalize these ideas with the following definitions, that deal with a physical graph G and a graph of requests I.

2.1.2.1 Defining an orientation

In the case when $G = \vec{C}_N$, although we have said that we get rid of the orientation of the arcs because of the symmetry, it will be necessary to define an orientation of the requests in a subgraph B_ω, to have no ambiguity in the definition of l_G and l_I. Let B_ω be a subgraph on vertices $a_1, a_2, \ldots, a_{|B_\omega|}$.

Suppose that $|B_\omega| = 2q+1$ is odd. Wlog, consider the vertex a_1. Relabeling the vertices, we can suppose that $a_1 < a_2 < \ldots < a_{|B_\omega|}$. Then, partition the remaining vertices into 3 sets :

$$\mathcal{R} = \{a_2, \ldots, a_q\},\ \mathcal{M} = \{a_{q+1}, a_{q+2}\},\ \mathcal{L} = \{a_{q+3}, \ldots, a_{2q+1}\}$$

We can see an example in Figure 2.3.

Now define the following orientation for a general request between a pair of vertices i and j :

- If $i = a_1$ and $j \in \mathcal{R}$, orient the request (a_1, j)
- If $i = a_1$ and $j = a_{q+1}$, orient the request (a_1, a_{q+1})
- If $i = a_1$ and $j = a_{q+2}$, orient the request (a_{q+2}, a_1)
- If $i = a_1$ and $j \in \mathcal{L}$, orient the request (j, a_1)
- If $i \in \mathcal{R}$ and $j \in \mathcal{M}$, orient the request (i, j)
- If $i \in \mathcal{L}$ and $j \in \mathcal{M}$, orient the request (j, i)
- If $i \in \mathcal{R}$ and $j \in \mathcal{R}$ (and thus, $i < j$), distinguish two cases :
  - if $j - i \leq q$, orient the request (i, j)
  - if $j - i > q$, orient the request (j, i)
- If $i = a_{q+1}$ and $j = a_{q+2}$, orient the request (a_{q+1}, a_{q+2})

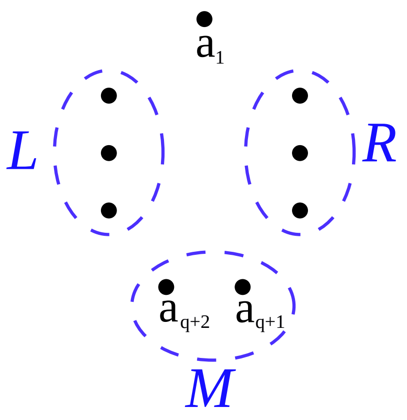


Fig. 2.3 – Partitioning the vertex set

Suppose now that $|B_\omega| = 2q$ is even. As before, consider a_1 and partition the remaining vertices into 3 sets :

$$\mathcal{R} = \{a_2, \ldots, a_q\},\ \mathcal{M} = \{a_{q+1}\},\ \mathcal{L} = \{a_{q+2}, \ldots, a_{2q}\}$$

Now define the following orientation for a general request between a pair of vertices i and j :

- If $i = a_1$ and $j \in \mathcal{R}$, orient the request (a_1, j)
- If $i = a_1$ and $j = a_{q+1}$, orient the request (a_1, a_{q+1}) or (a_{q+1}, a_1). Remark that it will be no always possible to choose between both orientations, as we can see in Figure 2.4 where, provided that $C = 3$, we are forced to orient the request (a_{q+1}, a_1) because the edges of $\mathcal{R}$ are already saturated
- If $i = a_1$ and $j \in \mathcal{L}$, orient the request (j, a_1)
- If $i \in \mathcal{R}$ and $j \in \mathcal{M}$, orient the request (i, j)
- If $i \in \mathcal{L}$ and $j \in \mathcal{M}$, orient the request (j, i)
- If $i \in \mathcal{R}$ and $j \in \mathcal{R}$ (and thus, $i < j$), distinguish three cases :
  - if $j - i < q$, orient the request (i, j)
  - if $j - i > q$, orient the request (j, i)
  - if $j - i = q$, orient the request (i, j) or (j, i)

Remark 2.1.4 *If $G = \vec{C}_N$ we can give the explicit formula of $l_{\vec{C}_N}$. Indeed, if we consider $i, j \in \mathbb{N}$:*

$$l_{\vec{C}_N}((i,j)) = \begin{cases} j - i, & \text{if } j > i \\ N + i - j, & \text{if } j < i \end{cases}$$

Although the previous formulation is correct and rigorous, it might be interesting to define an orientation (that will turn out to be the same) in a more easy an intuitive way. First of all, let's define a (non transitive) order in the vertex set of a graph.

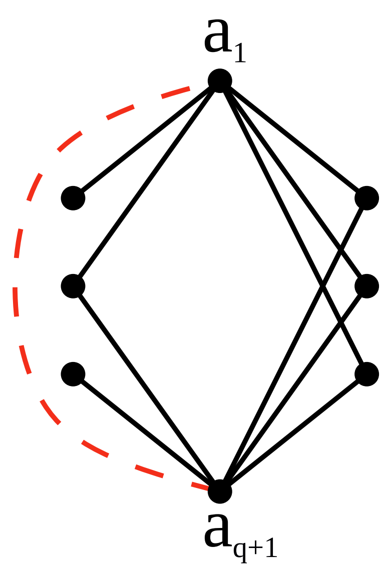


FIG. 2.4 – Giving an orientation to the requests

Definition 2.1.2 *Given a graph G with vertex set $V(G) = \{0, ..., N-1\}$ we will say that $i <_G j$ if $j - i \equiv x \pmod N$ with $x \in (0, \lfloor \frac{N}{2} \rfloor)$. If N is and $j - i \equiv \frac{N}{2} \pmod N$, then the order can be any of both possibilities ($i <_G j$ or $i >_G j$).*

And with this order we can define the following orientation :

Assign direction to the edges (i, j) if $i <_G j$.

2.1.2.2 Defining the lengths

Let's begin by defining the length l_G.

Definition 2.1.3 *Given a request (i, j), let l_G be the length of the request in the graph G, that is, the length of the shortest path going from i to j in G.*

Recall that in the case when G is a bidirectional ring, if we consider only one direction of the ring then we have to be careful with l_G ($l_{\vec{C}_N}$ in this case). For instance, if we deal with a ring on 9 nodes oriented clockwise, then the request $(4, 2)$ has length 7 rather than 2, since the shortest (and unique, in this case) path from 4 to 2 has length 7.

To properly define l_I the notion of *graph embedding* is needed.

Definition 2.1.4 (Graph embedding) *A* graph embedding *is a map from one graph (the* guest *graph) into another (the* host *graph), such that a guest graph* node *is assigned*

to a host graph node *and a guest graph* edge *is assigned to a host graph* path.
Three parameters are defined in a embedding :

- ***Node load*** *: maximum number of guest graph nodes mapped to the same host graph node. It is usually referred as* load, *but we are using this new notation to avoid confusion with the* load *defined before in this work.*
- ***Dilation*** *: length of the longest path in the host graph mapped by an edge in the guest graph edge.*
- ***Load*** *: maximum number of host graph path mapped by an edge in the guest graph edge that go through the same host graph edge. It is usually referred as* congestion, *but we have used this notation because it is exactly the same as the* load *defined before.*

Now we are able to define l_I :

Definition 2.1.5 *Given a request (i, j) in a subgraph B_ω of I, embed B_ω on a cycle with the same order $\overrightarrow{C}_{|B_\omega|}$. Now, let l_I (or l_{B_ω} if it is necessary to specify to which subgraph we refer) be the length of the shortest path going from i to j in $\overrightarrow{C}_{|B_\omega|}$.*

For instance, consider Figure 2.5, where a ring on 15 nodes is depicted. The full dots represent the vertices inside a certain subgraph, labeled as A, B, C, D, E, F, G. The other 8 nodes are in the ring but not in the subgraph. Thus, in this example, the request (B, C) has length l_I 1, while l_G is 3, and the request (F, A) has length l_I 2, while l_G is 6. Finally, the request (C, F) has length l_I 3, while l_G is 5.

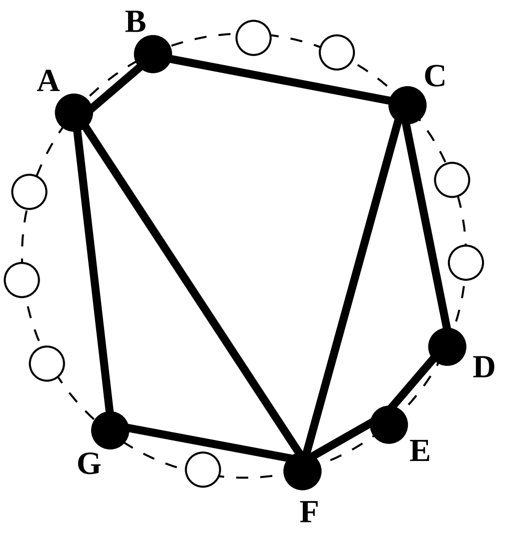


Fig. 2.5 – About the length of the requests in the ring

2.1.3 About embedding the subgraphs in the Ring

To understand to problem that we will study in this section, let's begin with an example. Let $C = 2$, and we all agree that the graph of Figure 2.6 is a priori a valid graph with grooming factor 2.

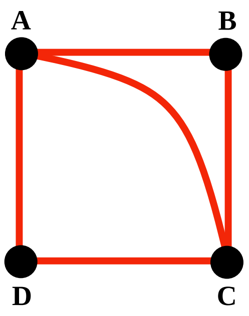


FIG. 2.6 – Example of a valid graph with grooming factor 2

Now let $N = 15$, and we have to embed this graph in the ring, i.e., we have to choose 4 of the 15 vertices to be A, B, C, D. Suppose that we embed the graph like it is shown in Figure 2.7, where the center of the ring is represented by a full blue dot. But here we see that we have load 3 in 4 arcs of the ring (those going from A until C), hence it is not a valid embedding with this grooming factor !

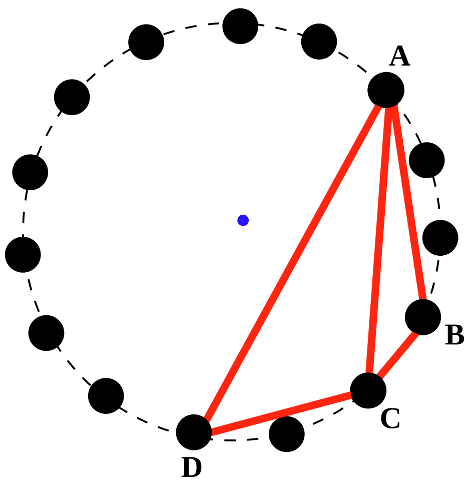


FIG. 2.7 – Example of a bad embedding with grooming factor 2

With this example we can see that it is a problem that we cannot stand in the way. Moreover, it might be possible than we could not choose any labeling of the vertices of the subgraphs in order to ensure the validity of the embedding. To convince ourselves that it can really occur in explicit decompositions, see in Remark 2.1.5 and Remark 2.1.6 that it is not always possible to do such an embedding.

Remark 2.1.5 *If C=1, and we partition K_9 into the C_4's $(i, 1+i, 5+i, 3+i, i)$, for $i = 0, \dots, 8$, then there exists no possible permutation of the labels of the vertices respecting the grooming factor.*

Proof: Let's suppose that there exists such a permutation. To ensure the embedding, the vertices of all the cycles must be ordered cyclically modulo N. In general, this permutation will be : $0 \rightarrow a$, $1 \rightarrow b$, $2 \rightarrow c$, $3 \rightarrow d$, $4 \rightarrow e$, $5 \rightarrow f$, $6 \rightarrow g$, $7 \rightarrow h$, $8 \rightarrow i$. Note than we can do the permutations $j \rightarrow (j+1)$ until we can choose 3 couples of vertices in the cycles (x_1, x_2) with $x_1 < x_2$ (i.e. without taking into account the congruence). It is because each cycle (after the permutation) has 4 couples of vertices, and in each cycle there is as much 1 couple of the form (x_1, x_2) with $x_1 > x_2$ (we can only "cross 0" as much once). Now, if all works we can choose the 3 couples and we have :

$f < a$, because we have the cycle (4,**5**,**0**,7,4)
$a < h$, because we have the cycle (4,5,**0**,**7**,4)
$h < f$, because we have the cycle (2,3,**7**,**5**,2)

But we have obtained $f < a < h < f \;\Rightarrow\; f < f$, that is a contradiction, and then we conclude that there exists no permutation. □

Remark 2.1.6 *If C=1, and we partition K_7 into the C_3's $(i, 1+i, 3+i, i)$, for $i = 0, \dots, 6$, then there exists no possible permutation of the labels of the vertices respecting the grooming factor.*

Proof: It is similar to the previous proof. □

Given an explicit decomposition of a complete graph into subgraphs, how can we know if it is possible to embed each subgraph into the original graph respecting the load constraint given by the grooming factor ?

The basic idea is to ensure that the "hole" of the subgraph coincides with the "big hole" of the ring. Defining the length always following the same direction in the ring (either clockwise or counterclockwise), a sufficient condition is that there are no requests of length l_G greater or equal than $\lfloor \frac{N}{2} \rfloor$, as we prove in Proposition 2.1.1. Recall that now by *length* of a request we mean the actual length in the ring $l_{\vec{C}_N}$, not inside the subgraph.

Proposition 2.1.1 *Suppose that we have a decomposition of T_N into subgraphs, where each subgraph satisfies the load constraint, and that we have a labeling of the vertices such that with this labeling there are no requests of length $l_{\vec{C}_N}$ greater or equal than $\lfloor \frac{N}{2} \rfloor$. Then, the embedding of the subgraphs in the ring satisfies the load constraint, and hence this is a valid solution of the problem.*

Proof: Wlog, suppose that the ring is oriented clockwise, and that the requests are oriented as it is described above. Then, think up that from the center of the circumference defined by the ring, we push all the requests of each subgraph until the circumference. For each request, two things can happen : either its direction is the same as the direction of the ring, or it is not. If all the directions of all the requests of a subgraph agree with the direction of the ring, then we will have the same load that we had in the subgraph. Given a request (i, j), the directions will coincide if $j < i + \lfloor \frac{N}{2} \rfloor \pmod N$. But, if there are no requests of length greater or equal than $\lfloor \frac{N}{2} \rfloor$, all the directions will coincide, and then by embedding the subgraphs in the ring we cannot increase the load more than C, because we are supposing that in the subgraphs the load constraint is satisfied. Hence, this is a valid solution. $\square$

The same idea can be expressed in a much more useful way by using the definition of the orientation based on the order relation.

Proposition 2.1.2 *Suppose that we have a decomposition of T_N into subgraphs B_ω, where each subgraph satisfies the load constraint. Suppose also that for every request $(a_i, a_j) \in E(B_w)$,*

$$i <_{B_w} j \Rightarrow a_i <_G a_j$$

Then, the embedding of the subgraphs in the ring satisfies the load constraint, and hence this is a valid solution of the problem.

Proof: If the orientations coincide for each request in each subgraph, we will have the same load in the ring than there was in the subgraphs, and thus the load constraint will be satisfied. $\square$

Now look again at Figure 2.7, but now suppose that the grooming factor is 3. In this case this is indeed a good embedding!
Let's try to capture this phenomenon, and with this aim the following definition arises.

Definition 2.1.6 *Given a request r on a subgraph B_ω, let*

$$\widehat{L}(r, B_\omega) := \max_{e \in G} L(e, B_\omega - r),$$

where $B_\omega - r$ is the graph obtained from B_ω by deleting the edge r.

This is, $\widehat{L}(r, B_\omega)$ is the maximum load in the arcs of G induced by all the requests of B_ω except r. Although usually the subgraphs that we will use to partition the request set will have the arcs saturated, it can be useful to allow some relaxation on the routing of the requests, and this is the idea of Proposition 2.1.3, that softens a bit the sufficient conditions of Proposition 2.1.1.

Proposition 2.1.3 *Suppose that we have a decomposition of T_N into subgraphs, where each subgraph B_ω satisfies :*

1) $L(e, B_\omega) \leq C, \ \forall e \in G$

2) $\forall r \in E(B_\omega)$ *such that* $l_G(r) \geq \lfloor \frac{N}{2} \rfloor \ \Rightarrow\ \widehat{L}(r, B_\omega) < C$

Then, the embedding of the subgraphs in the ring satisfies the load constraint, and hence this is a valid solution of the problem.

Proof: The first condition is just the load constraint in each subgraph. In the second condition we are allowing requests of l_G greater or equal than $\lfloor \frac{N}{2} \rfloor$. But what this condition guarantees is that, for each request r such that $l_G(r) \geq \lfloor \frac{N}{2} \rfloor$, then $\widehat{L}(r, B_\omega)$ is *strictly* lower than C, and thus we will not surpass the load by routing this request r. For all the other requests, Proposition 2.1.1 holds. □

2.2 Lower Bounds

2.2.1 $\gamma(C, p)$

To obtain accurate lower bounds we need to bound the value of $|E_w|$ for a graph with $|V_w| = p$ vertices, satisfying the load constraint.

Definition 2.2.1 *Let* $\gamma(C, p)$ *be the maximum number of edges of any graph* $H = (V, E)$ *with* $|V| = p$*, such that* $L(H, e) \leq C$, $\forall e \in E(H)$*. Recall that the load constraint is always assumed to be defined in the ring.*

The determination of $\gamma(C, p)$ was a challenging problem. We have solved it in Proposition 2.2.1 of Section 2.1.2.

Equations (2.2) and (2.3) become

$$A = \sum_{p=2}^{N} p a_p \qquad (2.4)$$

$$\sum_{p=2}^{N} a_p \geq \left\lceil \frac{N^2 - \varepsilon}{8C} \right\rceil, \text{ where } \varepsilon = \begin{cases} 0, & \text{if } N \text{ even} \\ 1, & \text{if } N \text{ odd} \end{cases} \qquad (2.5)$$

$$\sum_{p=2}^{N} a_p \gamma(C, p) \geq \frac{N(N-1)}{2} \qquad (2.6)$$

We will see in this section that, in a Ring topology, the best we can do to groom the maximum number of requests in a graph with a given number of vertices (respecting the load constraint given by the grooming factor, of course) is to use the requests of minimum length l_I (1, 2, 3...) until we use all the requests, or until we exceed the load constraint. This property may seem very intuitive, but it needs to be proved, because in a Path topology it is not true, as we can see in the counterexample given in [4] and illustrated in Figure 2.8. In this example, $N = 11$ and $C = 10$. The nodes are numbered from 0 till 10. If we use all the requests of lengths 1, 2, 3 and 4, we obtain 34 requests, but if we replace the request $(3, 7)$ by the longer requests $(0, 5)$ and $(5, 10)$ we obtain 35 requests.

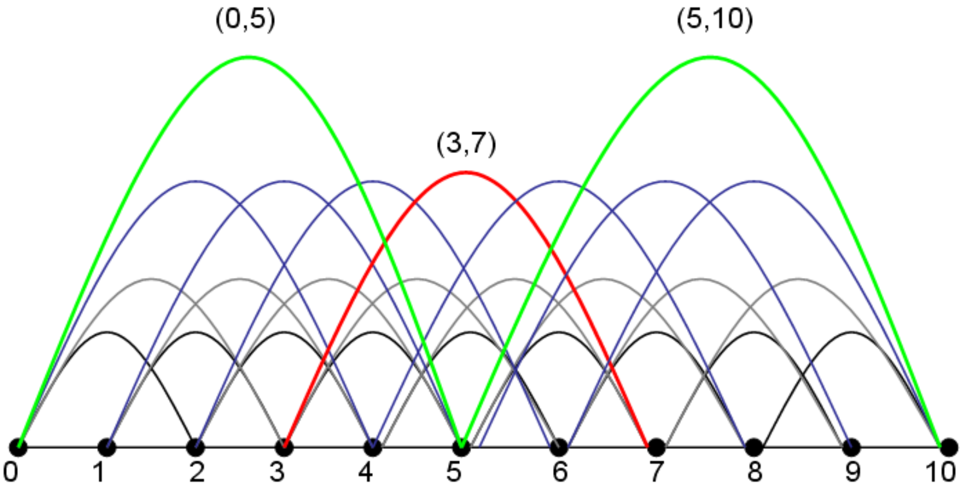


FIG. 2.8 – Counterexample on the Path

Happily, it doesn't happen in our case, as we will see below. In Proposition 2.2.1 (after stating a simple remark) we establish the value of $\gamma(C, N)$ for any value of C and N.

Remark 2.2.1 *For all $C \in \mathbb{N} \setminus \{0\}$, there exist $k \in \mathbb{N} \setminus \{0\}$ and $r \in \mathbb{N}$, $0 \le r < k+1$, such that*

$$C = \frac{k(k+1)}{2} + r$$

Proof: We only have to begin to do $1 + 2 + 3 + \dots$ until we surpass (strictly) C. Let L be the last value in the addition, therefore $1 + 2 + 3 + \dots + L - 1 + L > C$. Trivially, we have $k = L - 1$, and $r = N - \frac{k(k+1)}{2}$. Or, even faster, we can solve directly the equation in k and find $k = \left\lfloor \frac{-1+\sqrt{1+8C}}{2} \right\rfloor$ and then find r as before. □

Proposition 2.2.1 (Requests of Shortest Length) *$\gamma(C, p)$ is achieved by using the requests of shortest length l_I, until covering all the load. Moreover, if we write C in the form $C = \frac{k(k+1)}{2} + r$, with $0 \le r \le k$, then*

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

Proof: If $p \le 2k + 1 + \varepsilon$, with $\varepsilon = 1$ *if* $r \ge \frac{k+2}{2}$, we can put all the requests without overloading the arcs, and therefore $\gamma(C, p) = \frac{p(p-1)}{2}$.
If $p \ge 2k+2+\varepsilon$, with $\varepsilon = 1$ *if* $r \ge \frac{k+2}{2}$, we can have a solution by taking all the requests of length $1, 2, \dots, k$ plus $\left\lfloor \frac{rp}{k+1} \right\rfloor$ requests of length $k + 1$, giving

$$\gamma(C,p) \ge kp + \left\lfloor \frac{rp}{k+1} \right\rfloor \tag{2.7}$$

Let us show that this is the best solution. Wlog, take the requests clockwise. Suppose that we have any other solution with γ requests, γ_i being of length i. Then each request of length i loads exactly i arcs, and we have :

$$Cp \geq \sum_{i=1}^{\infty} i\gamma_i, \text{ but } \sum_{i=1}^{\infty} i\gamma_i \geq \sum_{i=1}^{k} i\gamma_i + (k+1)\left(\gamma - \sum_{i=1}^{k}\gamma_i\right) =$$

$$= \sum_{i=1}^{k} ip + (k+1)(\gamma - kp) + \sum_{i=1}^{k} \underbrace{(k+1-i)(p-\gamma_i)}_{\geq 0} \geq \frac{k(k+1)}{2}p + (k+1)(\gamma - kp)$$

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

$$\gamma \leq kp + \frac{rp}{k+1} \tag{2.8}$$

Combining (2.7) and (2.8), and taking into account that $\gamma(C,p) \in \mathbb{N}$, we obtain the result. $\square$

We shall give an alternative proof of Proposition 2.2.1.

Proof: [Alternative proof] We will suppose in the whole proof that p is large enough, to be able to use the longest requests. First of all, we will use that the result is true if C is of the form $C = 1 + 2 + \ldots + k = \frac{k(k+1)}{2}$. In this case, if we use all the requests of shortest length (from 1 till k) we have all the arcs with maximum load, and we cannot do it better because if we want to add 1 longer request we would have to remove at least 2 requests. So the property is easily true for these values of C.

$$C = \frac{k(k+1)}{2} \;\Rightarrow\; k \cdot p \textit{ requests} \;\Rightarrow\; \gamma\left(C, \frac{k(k+1)}{2}\right) = k \cdot p$$

$$C = \frac{(k+1)(k+2)}{2} \;\Rightarrow\; (k+1) \cdot p \textit{ requests} \;\Rightarrow\; \gamma\left(C, \frac{(k+1)(k+2)}{2}\right) = (k+1) \cdot p$$

Now let's see what happens for intermediate values of C. So let $C = \frac{k(k+1)}{2} + r$, with $r \leq k+1$.
Turning around, we have a solution, using all the requests of length $\leq k$ and as many requests of length (k+1) as we can, until covering the load, with

$$\widetilde{\gamma} = kp + \left\lfloor \frac{rp}{k+1} \right\rfloor \textit{ requests}$$

To prove that this solution is the best, consider any other solution with γ requests. In this new solution, we will have :

requests of length 1 $\Rightarrow$ at most p
requests of length 2 $\Rightarrow$ at most p
requests of length 3 $\Rightarrow$ at most p
$\vdots$
requests of length k $\Rightarrow$ at most p
requests of length k+1 $\Rightarrow$ at least $(\gamma - kp)$

Let´s add up all the lengths of all the requests. Note that the minimum total length will be achieved by taking as many requests as we can of lengths from 1 till k, i.e. p. So we have :

$$\sum length\ of\ all\ requests \geq p(1+2+\ldots+k)+(\gamma-Kp)(k+1)$$

Now suppose that the new solution γ is better than our solution $\widetilde{\gamma}$, for example $\gamma=\widetilde{\gamma}+1$, therefore

$$\sum length\ of\ all\ requests \geq N\frac{k(k+1)}{2}+\left(\left\lfloor\frac{rp}{k+1}\right\rfloor+1\right)(k+1)$$

If we divide this value by p, we will obtain the average load of the arcs :

$$average\ load \geq \frac{k(k+1)}{2}+\left(\left\lfloor\frac{rp}{k+1}\right\rfloor+1\right)\frac{(k+1)}{p}$$

We can write $\left\lfloor\frac{rp}{k+1}\right\rfloor=\frac{rp-\varepsilon}{k+1}$, with $\varepsilon<k+1$, and taking into account that $C=\frac{k(k+1)}{2}+r$ and that $\frac{k+1-\varepsilon}{p}>0$ we conclude that

$$average\ load \geq \underbrace{\frac{k(k+1)}{2}+r}_{C}+\underbrace{\frac{k+1-\varepsilon}{p}}_{>0}>C$$

that is a contradiction because we cannot have a load strictly greater than the grooming factor, and we conclude that there cannot exist any valid solution better than $\widetilde{\gamma}$, so we have finished.

□

2.2.2 $\rho(C)$

We will begin by defining a parameter, $\rho_{min}(C)$, that is strongly related to $\gamma(C,p)$, but that we will see that is helpful in some cases (see Section 2.7).

Definition 2.2.2 *Given a subgraph B_ω of I, we define $\rho(B_\omega)=\frac{|V(B_\omega)|}{|E(B_\omega)|}$. We can define the minimum ratio depending on grooming factor C.*

$$\rho_{min}(C)=\min\{\rho(B_\omega)|L(B_\omega,e)\leq C\ \forall e\in E(B_\omega)\}$$

Lemma 2.2.1

$$\frac{1}{\rho_{min}(C)}=\max_p\left(\frac{\gamma(C,p)}{p}\right) \tag{2.9}$$

Proof: By definition, $\max_p\left(\frac{\gamma(C,p)}{p}\right)=\max_p\left(\frac{|E|}{p}\right)\Rightarrow\frac{1}{\rho_{min}(C)}=\max_p\left(\frac{|E|}{p}\right)=\max_p\left(\frac{\gamma(C,p)}{p}\right)$
□

Proposition 2.2.2 *If $C=\frac{k(k+1)}{2}$, $\rho_{min}(C)$ is achieved by the complete graph K_{2k+1} and by the complete multipartite graph $K_{2x(k+1)}$ and then $\rho_{min}(\frac{k(k+1)}{2})=\frac{2k+1}{\frac{2k(2k+1)}{2}}=\frac{1}{k}$*

Proof: If $C = \frac{k(k+1)}{2}$ then, applying Theorem 2.2.1 we deduce that $\gamma(\frac{k(k+1)}{2}, p) = k \cdot p$. Now, using (2.9) we have that $\frac{1}{\rho_{min}(\frac{k(k+1)}{2})} = \max_p \left(\frac{\gamma(\frac{k(k+1)}{2}, p)}{p} \right) = k$.
Using Proposition 2.2.1 it is easy to deduce that the complete graph K_{2k+1} and the complete multipartite graph $K_{2x(k+1)}$ achieve the $\rho_{min}(C)$, because both of them use the requests of shortest length. □

We can generalize the previous result to any value of C, as we will see in the next proposition.

Proposition 2.2.3 *Writing C in the form $C = \frac{k(k+1)}{2} + r$, with $r \leq k+1$, then*

$$\rho_{min}(C) = \frac{k+1}{k(k+1)+r} \tag{2.10}$$

Proof: From Theorem 2.2.1 we know that $\frac{\gamma(C,p)}{p} = k + \frac{1}{p} \left\lfloor \frac{rp}{k+1} \right\rfloor$, therefore

$$\max_p \left(\frac{\gamma(C,p)}{p} \right) = k + \frac{r}{k+1} = \frac{1}{\rho_{min}(C)} \;\Rightarrow\; \rho_{min}(C) = \frac{k+1}{k(k+1)+r}$$

□

2.2.3 General Lower Bounds

Theorem 2.2.1 (General Lower Bound) *The number of ADMs required in a bidirectional ring with N nodes and grooming factor C is lower bounded by the expression*

$$A(C,N) \geq \left\lceil \frac{N(N-1)}{2} \frac{k+1}{k(k+1)+r} \right\rceil \tag{2.11}$$

Where we have written $C = 1 + 2 + \ldots + (k-1) + k + r = \frac{k(k+1)}{2} + r$, with $r < k+1$.

Proof: As we have seen in Proposition 2.2.1, $\gamma(C,p) = kp + \left\lfloor \frac{rp}{k+1} \right\rfloor$, and using this in (2.6) we can write :

$$\frac{N(N-1)}{2} \leq \sum_{p=2}^{N} a_p \gamma(C,p) = \sum_{p=2}^{N} a_p \left(kp + \left\lfloor \frac{rp}{k+1} \right\rfloor \right) = \{using\ (2.4)\} =$$

$$= k \cdot A(C,N) + \sum_{p=2}^{N} a_p \left\lfloor \frac{rp}{k+1} \right\rfloor \;\Rightarrow$$

$$\Rightarrow\; \frac{N(N-1)}{2} - k \cdot A(C,N) \leq \sum_{p=2}^{N} a_p \left\lfloor \frac{rp}{k+1} \right\rfloor \leq \sum_{p=2}^{N} a_p \frac{rp}{k+1} = \frac{r}{k+1} A(C,N) \;\Rightarrow$$

$$\Rightarrow\; A(C,N) \cdot (k + \frac{r}{k+1}) \geq \frac{N(N-1)}{2} \;\Rightarrow$$

$$\Rightarrow \; A(C,N) \geq \frac{N(N-1)}{2}\frac{1}{k+\frac{r}{k+1}} = \frac{N(N-1)}{2}\frac{k+1}{k(k+1)+r} \quad ,$$

as claimed. □

<u>Examples</u>

- $C = 1 \;\Rightarrow\; k = 1,\; r = 0 \;\Rightarrow\; A(1,N) \geq \frac{N(N-1)}{2}\frac{2}{2} = \frac{N(N-1)}{2}$
- $C = 2 = 1+1 \;\Rightarrow\; k = 1,\; r = 1 \;\Rightarrow\; A(2,N) \geq \frac{N(N-1)}{2}\frac{2}{2+1} = \frac{N(N-1)}{3}$
- $C = 3 = 1+2 \;\Rightarrow\; k = 2, r = 0 \;\Rightarrow\; A(3,N) \geq \frac{N(N-1)}{2}\frac{3}{2\cdot 3} = \frac{N(N-1)}{4}$

We will obtain these values later looking directly at the equations, and trying to improve them. We will also see that sometimes it is possible to attain these lower bounds for specific values of C and N, and sometimes it is not.

In terms of the parameter $\rho(C)$, we have the next result that also determines us a lower bound of our problem :

Proposition 2.2.4 *Any grooming of R requests with grooming factor C needs at least* $R \cdot \rho_{min}(C)$ *ADMs and then* $A(C,N) \geq \frac{N(N-1)}{2}\rho_{min}(C)$.

Proof: We have

$$R = \sum_{\omega=1}^{W} |E(B_\omega)| \leq \frac{1}{\rho_{min}(C)} \sum_{\omega=1}^{W} |V(B_\omega)|$$

Then, $A(C,N) = \sum_{\omega=1}^{W} |V(B_\omega)| \geq R \cdot \rho_{min}(C) = \frac{N(N-1)}{2}\rho_{min}(C)$ □

Since $\rho_{min}(C) = \frac{k+1}{k(k+1)+r}$, note that we have given here in Proposition 2.2.4 an alternative (and easier) proof of Theorem 2.2.1.

One can thing in trying to improve a bit the previous lower bound, in the following way :
First, remember that $\left\lfloor \frac{rp}{k+1} \right\rfloor = \frac{rp-\varepsilon_p}{k+1}$ (with $\varepsilon < k+1$), were we have remarked that ε depends on p (in fact $\varepsilon_p = rp \;(mod\; k+1)$). Then, doing the same that we have done in the proof, we obtain :

$$A(C,N) \geq \frac{k+1}{k(k+1)+r} \left(\frac{N(N-1)}{2} + \underbrace{\frac{1}{k+1}\sum_{p=2}^{N} \varepsilon_p a_p}_{>0\ ?} \right) \tag{2.12}$$

At first sight (2.12) seems to be an improved lower bound, but the problem is that we cannot ensure that the second term of the equation is *strictly* greater than zero. In fact,

we can only ensure that $\sum_{p=2}^{N} \varepsilon_p a_p \geq 0$, so we cannot say anything better than (2.11) at this moment.
For instance, it is possible that, for a given solution, $\exists j$ such that $\varepsilon(j) = 0$ (i.e., such that $rj \equiv 0 \pmod{k+1}$), that $a_i = 0 \; \forall i \neq j$, and that $a_j = \frac{N^2-1}{8C}$. In this situation we would have that $\sum_{p=2}^{N} \varepsilon_p a_p = 0$.

2.2.3.1 Comparison of existing lower bounds

In [14] the Ring Grooming Problem in the bidirectional case is studied, without restrictions on the routing. In this article the authors state a new lower bound (regardless of the routing) and give a general constant factor $12\sqrt{2}$-approximation algorithm for a general set of requests.
First of all, we shall remember the previous lower bound which had been stated before [14]. Consider a general set of requests, and let d_{jk} be the amount of traffic demands from node j to node k. Note by $\widetilde{A}(C,N)$ the optimal number of ADMs used in the bidirectional ring on N nodes and grooming factor C, without restrictions on the routing. The very first lower bound is obtained directly from the LP formulation of the problem. We give to this lower bound the same name than it is given in [14].

Theorem 2.2.2 (Add/drop lower bound, [14])

$$\widetilde{A}(C,N) \geq \sum_{j=1}^{N} \left\lceil \sum_{k=1}^{N} \frac{d_{jk}}{2C} \right\rceil$$

In [30] the authors give another lower bound which is sometimes better that the add/drop lower bound. Some additional notation is needed, following the same useful notation. Let q_{jk} and r_{jk} be the quotient and remainder when d_{jk} is divided by C, that is :

$$d_{jk} = Cq_{jk} + r_{jk}, \; \textit{with } 0 \leq r_{jk} < C$$

Order the d_{jk} with $j < k$ in such a way that their corresponding remainders r_{jk} decrease monotonically. We write D_p, Q_p and R_p, respectively, for d_{jk}, q_{jk} and r_{jk}, where p runs from 1 to $\frac{N(N-1)}{2}$ and the labeling is chosen so that $R_p \geq R_{p'}$, whenever $p < p'$. Now we are able to state the next lower bound.

Theorem 2.2.3 ([30]) *Suppose we are given an instance of the ring grooming problem. With the notation above, let P be the smallest integer such that*

$$\left(\sum_{p=1}^{\frac{N(N-1)}{2}} CQ_p \right) + \left(\sum_{p=1}^{P} \frac{R_p + C}{2} \right) \geq \sum_{p=1}^{\frac{N(N-1)}{2}} D_p.$$

Then, regardless of the routing, the minimum number of ADMs must satisfy

$$\widetilde{A}(C,N) \geq P + \sum_{p=1}^{\frac{N(N-1)}{2}} Q_p$$

In [14] another lower bound is stated, when $d_{jk} \neq 0$ for all $j \neq k$. In this case we have that $d_{jk}/d_{j'k'} \leq K$ for all $j \neq k$, $j' \neq k'$. This kind of traffic is called *K-quasi-uniform.* We set $d := max_{j,k}\{d_{jk}\}$, and we have that $d > 0$. Uniform traffic corresponds to the case $K = 1$, and then we can set $d = d_{jk}$, for any $j \neq k$, because all of them are equal.

Theorem 2.2.4 ([14]) *If traffic instance of ring grooming is K-quasi-uniform then, regardless of routing,*

$$\widetilde{A}(C,N) \geq \frac{(N^2-1)}{4}\sqrt{\frac{d}{2CK}}$$

In [14], at the end of Section 6, the authors talk about an improvement of this lower bound in the all-to-all unitary case, that is, $d = 1$, which is the case that we have studied. We would like to thank the authors for sending us by e-mail the details of the "factor-of-2 improvement" proof, which is only sketched in their article. We write this proof here in detail, but we recall that the arguments have been completely given by T.Chow and P.Lin.

Theorem 2.2.5 (T.Chow and P.Lin, by personal communication) *If a traffic instance of ring grooming is uniform and unitary, then, regardless of routing,*

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

Proof: The main idea of the proof is to lower bound the "wasted capacity" of the ADMs that are used in any solution. Note by A the number of ADMs used in this considered generic solution. Let's go now through the details.
If an ADM is supporting (in a generic solution) the termination of t traffic connections, that means that there must be t other ADMs on the same ring (i.e., on the same wavelength). Of those t connections, at most 2 of them are direct (to the ADMs beside it), and the other connections must pass through at least one other ADM. This is the idea of "wasted capacity". In fact, we can lower bound this waste : the next two shortest connections must pass through at least 1 ADM, the next 2 must pass through at least 2 ADMs, ... Summing all this, we obtain (wlog, suppose that t is even) : $2(1+2+\ldots+\frac{t}{2}-1) = 2(\frac{t}{2}-1)(\frac{t}{2})/2 = \frac{t^2}{4} - \frac{t}{2}$, which is the total wasted capacity contributed by this one ADM. The total waste is obtained just by summing this over all ADMs and divided by two (because each connection is counted twice, once at each termination). Thus, the total waste is given by

$$\frac{1}{2}\sum_{i=1}^{A}\left(\frac{t_i^2}{4} - \frac{t_i}{2}\right)$$

The choice of t_i for each ADM will determine the total waste, but there is a constraint on the t_i's based on the total number of terminations : $\sum_{i=1}^{A} t_i = N(N-1)$. We want to minimize the total waste under this constraint, and it is known that the best way to minimize the sum of the squares is to divide the sum into each component. This yields $t_i := \frac{N(N-1)}{A}$. Consequently, the minimum total waste is

$$\frac{1}{2}\sum_{i=1}^{A}\left(\frac{\left(\frac{N(N-1)}{A}\right)^2}{4} - \frac{\frac{N(N-1)}{A}}{2}\right) = \frac{N^2(N-1)^2}{8A} - \frac{N(N-1)}{4A}$$

Since the capacity of an ADM is C (C through connections, but up to 2C terminations), the total capacity of all ADMs is $C \cdot A$, which must necessarily be greater than the total waste. Therefore, we have

$$C \cdot A \geq \frac{N^2(N-1)^2}{8A} - \frac{N(N-1)}{4A} \quad \Longrightarrow \quad A \geq \frac{1}{2\sqrt{C}}\sqrt{\frac{N^2(N-1)^2}{2} - N(N-1)}\ ,$$

as claimed. □

Observe that, asymptotically :

$$\frac{1}{2\sqrt{C}}\sqrt{\frac{N^2(N-1)^2}{2} - N(N-1)} \ \approx\ \frac{N^2}{2}\frac{1}{\sqrt{2C}}$$

Therefore, as it was said in [27], this indeed means a factor-of-2 improvement with respect to the general lower bound stated in Theorem 2.2.4, which yields in this case $\frac{N^2-1}{4}\frac{1}{\sqrt{2C}}$.

Note by LB_1 the lower bound of Theorem 2.2.1 and by LB_2 the lower bound of Theorem 2.2.5. In order to simplify the computations, we compare them when $C = \frac{k(k+1)}{2}$:

$$\frac{LB_1}{LB_2} = \frac{N(N-1)}{2k}\frac{\sqrt{2k(k+1)}}{\sqrt{\frac{N^2(N-1)^2}{2} - N(N-1)}} > \frac{N(N-1)}{2k}\frac{\sqrt{2k^2}}{\sqrt{\frac{N^2(N-1)^2}{2} - N(N-1)}} =$$

$$= \frac{N(N-1)}{\sqrt{2}}\frac{1}{\sqrt{\frac{N^2(N-1)^2}{2} - N(N-1)}} > \frac{N(N-1)}{\sqrt{2}}\frac{1}{\sqrt{\frac{N^2(N-1)^2}{2}}} = 1 \ \Rightarrow\ LB_1 > LB_2$$

Thus, we conclude that, in the case of shortest path symmetric routing, $LB_1 > LB_2$. Hence, we have improved the lower bound.

2.3 Upper Bounds and Approximations

The main result that we will use to state our upper bound is this Erdös' theorem in the context of combinatorial designs, which is an upper bound for the *covering number* $c(v,l,t)$, this is, the minimum number of blocks in any $t-(v,l,\lambda)$ covering.

Theorem 2.3.1 (P. Erdös and J. Spencer, [26])

$$c(v,l,t) \le \left[\frac{\binom{v}{t}}{\binom{l}{t}}\right]\left[1+\log\binom{l}{t}\right]$$

Conjecture 2.3.1 (Upper Bound) *If N is odd and $C=\frac{k(k+1)}{2}+r$, with $r \le k$, the number of ADMs required in a bidirectional ring with symmetric shortest path routing is upper bounded by the expression*

$$A(C,N) \le \frac{N(N-1)}{2}\frac{1+\log[k(2k+1)]}{k}$$

Thus, we have an approximation with factor

$$\left(1+\frac{1}{\sqrt{C}}\right)(1+\log(4C))$$

Proof: Let $N=2q+1$ and write C in the usual form $C=\frac{k(k+1)}{2}+r$. It is not difficult to see that a complete subgraph K_{2k+1} with all the consecutive vertices in the subgraph verifying $dist(i,i+1) \le q$ satisfies the load constraint. Thus, a possible upper bound can be found by decomposing the edges of a K_{2q+1} into K_{2k+1}'s. In this situation the result of Theorem 2.3.1 is extremely useful. In our problem we have that $v=N=2q+1$, $l=2k+1$ and $t=2$. In this case, the upper bound says that

$$c(N,2k+1,2) \le \frac{N(N-1)}{(2k+1)2k}\left[1+\log\left(\frac{(2k+1)(2k)}{2}\right)\right]$$

Hence, we can upper bound the number of ADM's just by counting the number of vertices of each graph and the number of graphs :

$$A(C,N) \le (2k+1)\frac{N(N-1)}{(2k+1)2k}\left[1+\log\left(\frac{(2k+1)(2k)}{2}\right)\right] = \frac{N(N-1)}{2}\frac{1+\log[k(2k+1)]}{k}$$

Now we can compare this value with the lower bound of Theorem 2.2.1, obtaining in this value the claimed approximation ratio :

$$\frac{Upper\ Bound}{Lower\ Bound} = \frac{\frac{N(N-1)}{2}\frac{1+\log[k(2k+1)]}{k}}{\left\lceil\frac{N(N-1)}{2}\frac{k+1}{k(k+1)+r}\right\rceil} \le \frac{\frac{N(N-1)}{2}\frac{1+\log[k(2k+1)]}{k}}{\frac{N(N-1)}{2}\frac{k+1}{k(k+1)+r}} =$$

$$= \frac{(k(k+1)+r)\,(1+\log[k(2k+1)])}{k(k+1)} = \left(1+\frac{r}{k(k+1)}\right)(1+\log[k(2k+1)]) \le$$

$$\leq \left(1+\frac{1}{k}\right)(1+\log(4C)) \leq \left(1+\frac{1}{\sqrt{C}}\right)(1+\log(4C)),$$

recalling that we have used in the last step that $k \geq \sqrt{C}$, provided that $k \geq 4$.
In the case when N is even, we have to take care about the longest requests, but a similar result can be obtained with slight changes.
The only thing that remains to be checked is that there exists a covering such that the load constraint is satisfied. It seems to be a difficult but solvable problem.

□

2.4 Case C=1

For $C = 1$, $\gamma(1, p) = p$ if $p \geq 2$.

Using a result of the article written by *J-C. Bermond, L. Chacon, D. Coudert* and *F. Tillerot* [6] we get the optimal solution for $C = 1$ where the number of ADMS is $\frac{N(N-1)}{2}$.
Let's recall this results and some previous necessary definitions.
Given any pair of requests of a subgraph I_k of the covering of the logical graph I, we require that there should exist *edge disjoint routing* in G i.e. the paths associated in G to any pair of requests in the same subgraph must be edge disjoint. We call this property *the disjoint routing constraint (DRC)*. Such a covering will be called *DRC-covering* of K_n.
Denote by $\rho(n)$ the minimum number of cycles needed in a DRC-covering of K_n.

Theorem 2.4.1 ([6]) *When $n = 2p + 1$, $\rho(n) = \frac{p(p+1)}{2}$. Furthermore, the DRC-covering of K_{2p+1} consists of p C_3's and $\frac{p(p-1)}{2}$ C_4's.*

Theorem 2.4.2 ([6]) *When $n = 2p$, $p \geq 3$, $\rho(n) = \left\lceil \frac{p^2+1}{2} \right\rceil$. Furthermore, when $n = 4q$, the DRC-covering of K_{4q} consists of 4 C_3's and $2q^2 - 3$ C_4's, and when $n = 4q + 2$ the DRC-covering of K_{4q+2} consists of 2 C_3's and $2q^2 + 2q - 1$ C_4's.*

For $C = 1$ the best we can do is partition the set of requests into cycles, and this is what these covering result state. More precisely, the previous results prove that a complete graph can always be decomposed into C_3's and C_4's. For instance, in Figure 2.9 we can see a covering of K_{10}.

Indeed one can check that, in any of these coverings, all the C_3's and C_4's satisfy the distance conditions. For instance, let's check the case $N = 2q$. In [6] the proof is done by induction adding two new vertices, and the vertices are labeled as $A, 0, 1, \ldots, p-1, B, p, \ldots, 2p-1$. The C_4's are of the form $(A, i, B, p+i, A)$ and $(A, p+i, B, i, A)$, $0 \leq i \leq p-1$, and the C_3's are $(A, 0, B, A)$ and (B, p, A, B). Thus, we have no problem to embed these subgraphs.

Nevertheless, we can deduce the same result applying the procedure described in Section 2.7.1. On can check that this construction yields a partition into $K_3 = C_3$'s and C_4's, that works for all the odd values of N. Furthermore, we have already proved that the distance condition in satisfied with this construction.

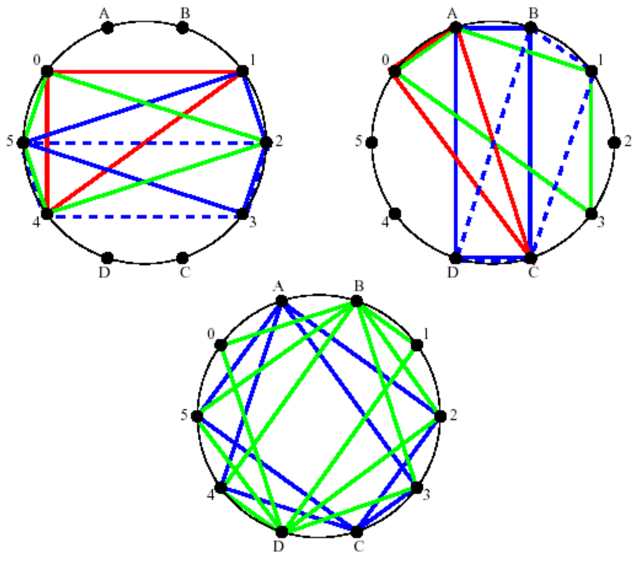


FIG. 2.9 – Cycles involved in the covering of K_{10}

2.5 Case C=2

2.5.1 Tighter Lower Bounds

For $C = 2$ we have that $\gamma(2,2) = 1$; $\gamma(2,3) = 3$; $\gamma(2,4) = 5$ and $\gamma(2,p) = \left\lfloor \frac{3p}{2} \right\rfloor$.

Note that such a solution is obtained if we call the vertices X_i, $1 \leq i \leq p$, by taking the requests X_i, X_{i+1} and X_i, X_{i+2} (indices modulo p) and supposing that their distance is at most $q = \frac{N-1}{2}$.

Equation (2.6) becomes

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

Therefore :

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

To attain the value $\frac{N(N-1)}{3}$ one would need to use decomposition with $G_6, G_8, G_{10}, , \ldots$ (by G_i we note a graph with i vertices) but that is impossible since by Equation (2.5) we will have $\frac{N^2-1}{16}$ subgraphs each with at least 6 vertices and therefore $A \geq \frac{6}{16}(N^2-1)$ and $\frac{6}{16} > \frac{1}{3}$.

We will see in a few lines that another tighter lower bound is $\left\lceil \frac{11N(N-1)}{32} \right\rceil$. Note that $\frac{11N(N-1)}{32} > \frac{N(N-1)}{3}$, so that means an actual improvement of the lower bound. The idea to look for this lower bound is the following intuitive informal but intuitive argument :

Think about an optimal solution, and guess what would be the optimum number of vertices of the subgraphs. Suppose also that most of the subgraphs have the same number of vertices, as it occurs frequently in known decompositions. Recall that $W \geq \frac{N^2-1}{16}$, $E = \frac{N(N-1)}{2}$ and that, if $p \geq 5$, $\gamma(2,p) = \lfloor \frac{3p}{2} \rfloor$. In order to find the optimum p, we solve the equation that equals the total number of requests :

$$\frac{N^2-1}{16} \cdot \frac{3p}{2} = \frac{N(N-1)}{2}$$

and we obtain that

$$p = \frac{16}{3}\frac{N(N-1)}{N^2-1} \overset{\mathrm{N}\to\infty}{\longrightarrow} \frac{16}{3} = 5,\hat{3}$$

Thus, the optimal graphs will have either 5 or 6 vertices. If we had a solution like this, we would have that

$$\begin{cases} a_5 + a_6 \geq \frac{N^2-1}{16} \\ 7a_5 + 9a_6 = \frac{N(N-1)}{2} \end{cases}$$

Roughly $\begin{cases} 8a_5 + 8a_6 \approx \frac{N^2}{2} \\ 7a_5 + 9a_6 \approx \frac{N^2}{2} \end{cases}$, and we obtain that $a_5 \approx a_6 \approx \frac{N^2}{32}$, and $A(2,N) = 5a_5 + 6a_6 \approx \frac{11}{32}N^2$.

Now we give a formal proof of the previously guessed lower bound.

Proposition 2.5.1 (Tighter Lower Bound for C=2) *The number of ADMs required in a bidirectional ring with N nodes and grooming factor 2 is lower bounded by the expression*

$$A(2,N) \geq \left\lceil \frac{11N(N-1)}{32} \right\rceil \tag{2.13}$$

Proof: We want to find a lower bound for the number of ADMs, that we call A. We have :

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

$$\left(\textit{using that } \frac{2}{3}\sum_{p=2}^{N} a_p\gamma(2,p) \geq \frac{N(N-1)}{2} \right) \geq \frac{N(N-1)}{3} + \frac{4}{3}a_2 + a_3 + \frac{2}{3}a_4 + \frac{1}{3}(a_5 + a_7 + a_9 + \ldots)$$

As we have seen before, we cannot use only graphs with 6, 8, 10... vertices, and so $\frac{N(N-1)}{3}$ is unreachable. Because of the same contradiction, we cannot use graphs with more than 6 vertices, i.e. 7, 9... Now, to increase the minimum as possible the lower bound, one can think that it would be a good idea to use some graphs with 5 vertices, because a_5 is the one with the lower coefficient ($\frac{1}{3}$) among a_2, a_3, a_4 and a_5. Anyhow, we can consider all these terms to handle the most general case.
So now we can write :

$$A \geq \frac{N(N-1)}{3} + \frac{a_5}{3} + \frac{2}{3}a_4 + a_3 + \frac{4}{3}a_2 \tag{2.14}$$

$$A \geq \left(\frac{N^2-1}{16} - a_5 - a_4 - a_3 - a_2\right) \cdot 6 + 5a_5 + 4a_4 + 3a_3 + 2a_2 \tag{2.15}$$

Now, since $N \geq 1 \Rightarrow N^2 - 1 \geq N(N-1)$ and therefore we can transform (2.15) into

$$A \geq \left(\frac{N(N-1)}{16} - a_5 - a_4 - a_3 - a_2\right) \cdot 6 + 5a_5 + 4a_4 + 3a_3 + 2a_2 =$$

$$= \frac{6}{16}N(N-1) - a_5 - 2a_4 - 3a_3 - 4a_2 \tag{2.16}$$

From (2.14) we get $-a_5 - 2a_4 - 3a_3 - 2a_2 \geq N(N-1) - 3A$, and using this in (2.16) we obtain

$$A \geq \frac{6}{16}N(N-1) + N(N-1) - 3A \quad \Rightarrow \quad 4A \geq (\frac{6}{16} + 1)N(N-1) \;\; \Rightarrow$$

$$\Rightarrow \; A \geq \frac{1}{4}\frac{6+16}{16}N(N-1) = \frac{11}{32}N(N-1),$$

as we wanted to see. The rounding to the nearest greater integer is because of the integrality of the number of ADMs. □

We can see in Table 2.1 the number of ADMs given by the Simplex and the Lower Bound for different values of N.

A graph of the ratio versus $\log(N)$ is drawn in Figure 2.10. We can observe in this graph that, at least "empirically", the ratio $\frac{ADM\ by\ Simplex}{Lower\ Bound} \overset{N\to\infty}{\longrightarrow} 1$, so we may think that we are in front of the right lower bound for C=2.

It is important to remark that the value of the number of ADMs that the Simplex gives may be impossible to achieve by an explicit construction. The reason of this phenomenon is that not all the constraints of the problem are taken into account in the Simplex formulation (load constraint, length of the requests, degree of the vertices, ...). Thus, this values are only useful to get an idea if our lower bounds are very far from the real ones, but they do not give at all a proof about the tightness of the bounds.

2.5.2 Upper bounds, constructions and approximations

The idea that we remark from the previous section is that the best we can do is to use graphs with 5 or 6 vertices. Let's try to make it more explicit by getting a bit into details. The fact that we would like to remark is that if we had an optimal solution, then we could have another one with only (or almost) graphs on 5 and 6 vertices, and with lower or equal drop cost.

N	$ADMs\ by\ Simplex$	$LowerBound : \left\lceil \frac{11}{32} N(N-1) \right\rceil$	Ratio $\frac{ADM\ Simplex}{Lower\ Bound}$
6	12	11	$1,090909091$
7	15	15	1
8	20	20	1
10	33	31	$1,064516129$
15	74	73	$1,01369863$
20	133	131	$1,015267176$
25	209	207	$1,009661836$
30	303	300	$1,01$
40	540	537	$1,005586592$
50	848	843	$1,005931198$
80	2180	2173	$1,003221353$
100	3413	3404	$1,002643948$
150	7698	7683	$1,001952362$
200	13700	13681	$1,001388787$
250	21425	21399	$1,00121501$
300	30865	30835	$1,00097292$
400	54900	54853	$1,000856836$
500	85819	85766	$1,00061796$
1000	343500	343407	$1,000270816$
2000	1374500	1374313	$1,000136068$
5000	8592500	8592032	$1,000054469$
10000	34372500	34371563	$1,000027261$
15000	77340000	77338594	$1,00001818$
20000	137495000	137493125	$1,000013637$
25000	214837500	214835157	$1,000010906$

TAB. 2.1 – Different values of the simulations compared with the Lower Bound for C=2

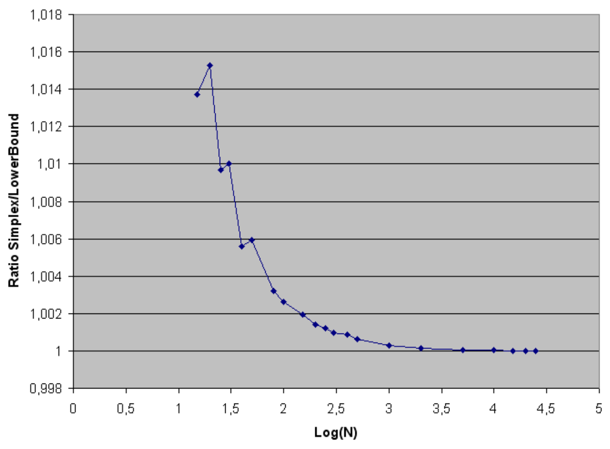


FIG. 2.10 – $\frac{ADM\ by\ Simplex}{Lower\ Bound}$ versus $\log(N)$ for $C = 2$ and $N \geq 15$

Remark 2.5.1 *If there is an optimal solution, then there is another optimal one with*

$$a_7 = a_8 = a_9 = \ldots = 0$$

Proof: Let's prove that we can get rid of of the graphs on 7 vertices. In Table 2.2 we see that we can use graphs having the same drop cost (number of vertices) but with more possible requests, and without graphs on 7 vertices. The same argument can be extended to any graph on more than 7 vertices.

Number of vertices	*Number of requests with* a_7	*Number of requests without* a_7
9	$a_7 + a_2 \longrightarrow 11$	$a_6 + a_3 \longrightarrow 12$
10	$a_7 + a_3 \longrightarrow 13$	$a_6 + a_4 \longrightarrow 14$
11	$a_7 + a_4 \longrightarrow 15$	$a_6 + a_5 \longrightarrow 16$
12	$a_7 + a_5 \longrightarrow 17$	$2a_6 \longrightarrow 18$

TAB. 2.2 – If we don't use graphs on 7 vertices, we can groom more requests without increasing the drop cost

$\square$

Remark 2.5.2 *If there is an optimal solution, then there is another optimal one with*

$$a_2 + a_3 + a_4 \leq 1$$

Vertices	*# of req. with* 2 *small graphs*	*# of req. with* 0 *or* 1 *small graph*
8	$a_4 + a_4 \rightarrow 10$	$a_6 + a_2 \rightarrow 11$
7	$a_4 + a_3 \rightarrow 8$	$a_6 + a_4 \rightarrow 9$
6	$a_4 + a_2 \rightarrow 6$	$a_6 \rightarrow 9$
6	$a_3 + a_3 \rightarrow 6$	$a_6 \rightarrow 9$
5	$a_3 + a_2 \rightarrow 4$	$a_5 \rightarrow 7$
4	$a_2 + a_2 \rightarrow 2$	$a_4 \rightarrow 5$

TAB. 2.3 – With one "small" graph is always enough

Proof: We can check in Table 2.3 that if we have 2 graphs on less than 5 vertices, then we can use only one "small" graph increasing the number of possible requests with the same drop cost.

□

Combining remarks 2.5.1 and 2.5.2 we can conclude that we can restrict us to solutions with graphs only on 5 and 6 vertices.
Hence, the graphs that we must use in the decompositions can be seen in Figures 2.11, 2.12 and 2.13.

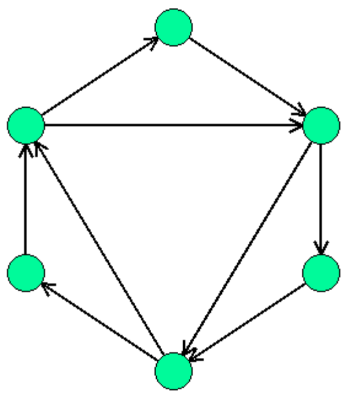


FIG. 2.11 – Graph with 6 vertices and 9 requests

2.5.2.1 $\frac{12}{11} = 1,0909$-approximation

Proposition 2.5.2 *Let N be odd, $N \geq 5$ and $C = 2$. Then we can find an approximation for the number of ADMs, with approximation ratio $\frac{12}{11}$.*

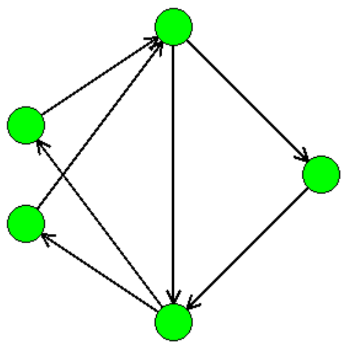


FIG. 2.12 – Graph with 5 vertices and 7 requests

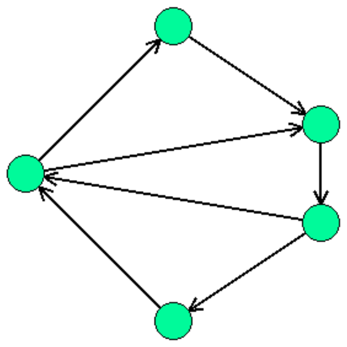


FIG. 2.13 – Another graph with 5 vertices and 7 requests

Proof: If $N = 5$ we use the following construction : let the vertex set be $\{0, 1, 2, 3, 4\}$. We use the subgraph on vertices $\{1, 3, 4\}$ with request set $\{13, 34, 41\}$, and the subgraph on vertices $\{0, 1, 2, 3, 4\}$ with request set $\{01, 12, 02, 23, 24, 30, 40\}$. In this way, we use 8 ADMs.
Now suppose that we have a valid construction for N, and we want to find a construction for $N + 2$. In the whole proof, we will use the graph in Figure 2.14 (not optimal).

Let $N = 2p + 1$, with p even. Let the vertex set be $\{a_0, a_1, \ldots, a_{p-1}, b_0, b_1, \ldots, b_{p-1}, \infty\}$. Let A and B the pair of new vertices that we want to join with the previous ones. Consider the graphs on vertices $\{A, a_i, a_{i\frac{p}{2}}, B, b_i, b_{i+\frac{p}{2}}\}$ and request set $\{Aa_i, Aa_{i+\frac{p}{2}}, a_iB, a_{i+\frac{p}{2}}B, Bb_i, Bb_{i+\frac{p}{2}}, b_iA, b_{i+\frac{p}{2}}A\}$, for $i = 0, \ldots, \frac{p}{2} - 1$. I.e., in each subgraph we join the two new vertices with 4 *old* vertices. Then, we use the graph on vertices $\{A, \infty, B\}$ to cover the requests $\{A\infty, \infty B, BA\}$.
Thus, each time we add two new vertices we use $\frac{p}{2} \cdot 6 + 3$ ADMs.

Let $N = 2p+1$, with p odd. Let the vertex set be $\{a_0, a_1, \dots, a_{p-1}, b_0, b_1, \dots, b_{p-1}, \infty\}$ as before. Let A and B the pair of new vertices that we want to join with the previous ones. Consider the graphs on vertices $\{A, a_i, a_{i+\frac{p-1}{2}}, B, b_i, b_{i+\frac{p-1}{2}}\}$ and request set $\{Aa_i, Aa_{i+\frac{p-1}{2}}, a_iB, a_{i+\frac{p-1}{2}}B, Bb_i, Bb_{i+\frac{p-1}{2}}, b_iA, b_{i+\frac{p-1}{2}}A\}$, for $i = 0, \dots, \frac{p-1}{2} - 1$. I.e., in each subgraph we join the two new vertices with 4 *old* vertices. Then, we use the graph on vertices $\{A, a_{p-1}, B, \infty, b_{p-1}\}$ to cover the requests $\{Aa_{p-1}, a_{p-1}B, AB, B\infty, Bb_{p-1}, \infty A, b_{p-1}A\}$.
Thus, each time we add two new vertices we use $\frac{p-1}{2} \cdot 6 + 5$ ADMs.

Observe that we have no requests of length greater than p, and thus we have no problems with the load constraint, because of the sufficient condition stated in Proposition 2.1.1.

To compute the number of ADMs of this construction, we have just to write the recurrence. Let's note by Z_p the number of ADMs when $N = 2p+1$. From the previous explanation, it is straightforward to realize that if $p \equiv 0 \pmod 2$, then $Z_p = Z_{p-2} + 6p - 4$. Thus, if $p \equiv 0 \pmod 2$, we have to solve :

$$\begin{cases} Z_p = Z_{p-2} + 6p - 4 \\ Z_2 = 8 \end{cases}$$

$$Z_p = Z_{p-2} + 6p - 4 = Z_{p-4} + 6(p-2) - 4 + 6p - 4 = \dots = \sum_{i=4, i \ even}^{p} (6i - 4) + 8 =$$

$$= \sum_{i=2}^{\frac{p}{2}} (12i - 4) + 8 = \dots = \frac{1}{8}(3N^2 - 2N - 1)$$

If $p \equiv 1 \pmod 2$ (i.e. $N \equiv 3 \pmod 4$)), we can write :

$$Z_p = Z_{p-1} + \frac{p-1}{2} \cdot 6 + 3 = \dots = \frac{1}{8}(3N^2 - 2N + 3)$$

Noting by $A_1(2, N)$ the number of ADMs of this construction, we can sum up :

- If $N = 2p+1$, $N \equiv 1 \pmod 4$, and $N \geq 5$, then :

$$A_1(2, N) = \frac{1}{8}(3N^2 - 2N - 1)$$

- If $N = 2p+1$, $N \equiv 3 \pmod 4$, and $N \geq 7$, then :

$$A_1(2, N) = \frac{1}{8}(3N^2 - 2N + 3)$$

In general, we can write :

$$A_1(2, N) = \frac{1}{8}(3N^2 - 2N - 1 + \varepsilon), \ where\ \varepsilon = \begin{cases} 0 & \text{if } N \equiv 1\ (mod\ 4) \\ 4 & \text{if } N \equiv 3\ (mod\ 4) \end{cases}$$

So we have that

$$\frac{A_1(2,N)}{\textit{Lower bound}} = \frac{\frac{1}{8}(3N^2 - 2N - 1 + \varepsilon)}{\frac{11}{32}N(N-1)} \overset{N\to\infty}{\longrightarrow} \frac{3}{8}\frac{32}{11} = \frac{12}{11}$$

We have found a $\frac{12}{11}$-approximation, as we wanted to see.

□

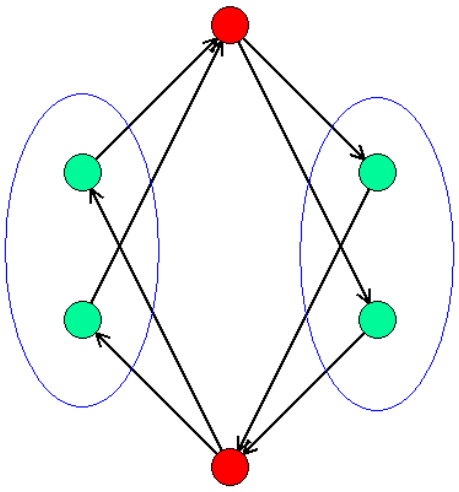


FIG. 2.14 – Graph with 6 vertices and 8 requests, used to match 2 new vertices with all the others (inside the circles)

2.5.2.2 $1,1103$-approximation

Idea : Partitioning the set of the nodes into 4 sets, and then use "optimal" graphs to match each set with the others, making in this way a *transversal construction*. Then, to cover the requests of each set we have plenty of possibilities, and one of them is using the optimal values for the Path (see [4]) ensuring that we have no problems with the length and the load constraint. Obviously, the fact of using the values from the Path implies a lack of optimality.

In this construction we have $N = 10p$ nodes divided into 4 sets, 2 of them with $3p$ elements, and the other ones with $2p$ elements, as we can see in Figure 2.15. The 10 sets of p elements are named $X', Y', Z', \alpha^1, \alpha^2, \beta^1, \beta^2, X, Y, Z$.

The goal is matching all the nodes among them, respecting the load constraint and using optimal graphs whenever possible. First of all, we match each element of each set with all the elements of all the other sets, using the optimal families of graphs of Figures 2.16,

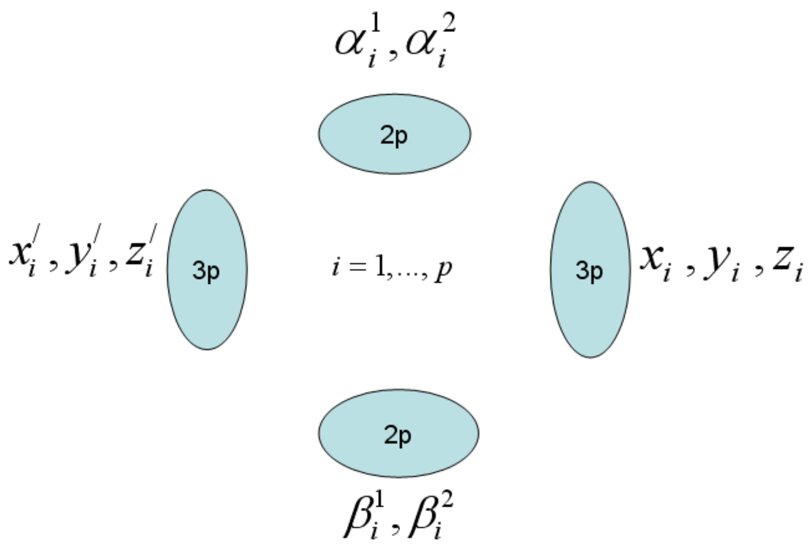


FIG. 2.15 – Configuration in the construction of Section 2.5.2.2

2.17, 2.18 and 2.19. One can check that using all these graphs we cover all the requests among any 2 sets. Since in each family of graphs $i, j = 1, \ldots, p$, we have $4p^2$ graphs, and thus $28p^2$ edges and $20p^2$ ADMs.

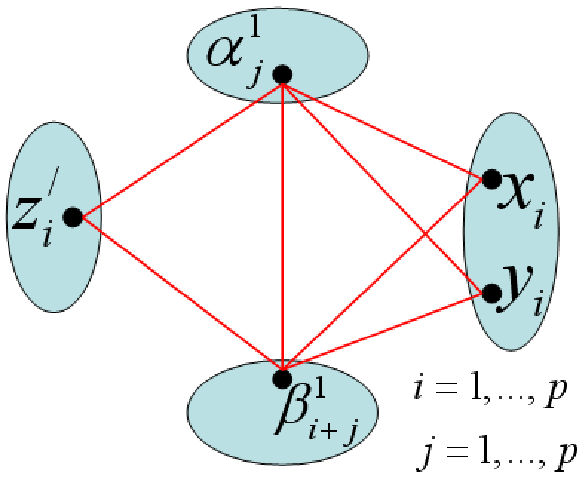


FIG. 2.16 – Graph used to match the vertices of the construction

Now what we have to do is matching among them the vertices of each of the 4 subsets. Suppose in this construction that p is even. In Figure 2.20 we can see the graphs used to match between them the couples $\{X, X'\}$, $\{X, Y'\}$, $\{Y, X'\}$, $\{Y, Y'\}$, $\{X, Z\}$, $\{X, Z'\}$, $\{Y, Z\}$, $\{Y, Z'\}$, $\{Z, Y'\}$, $\{Z, Z'\}$, $\{X', Y'\}$ and $\{X', Z'\}$. Adding up, we use in this way $9p^2$ ADMs.

Observe that we have no requests of length greater than $5p = \frac{N}{2}$, and thus we have

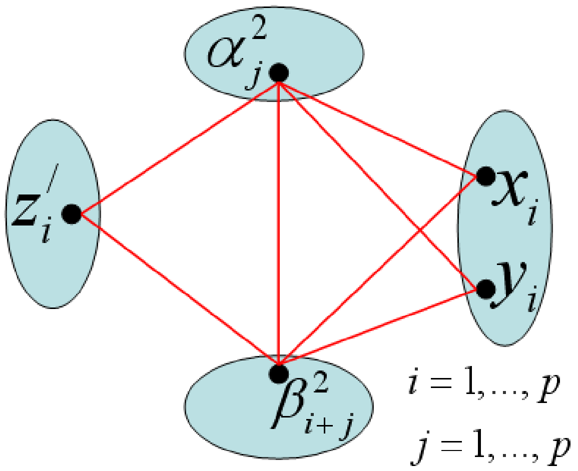


FIG. 2.17 – Graph used to match the vertices of the construction

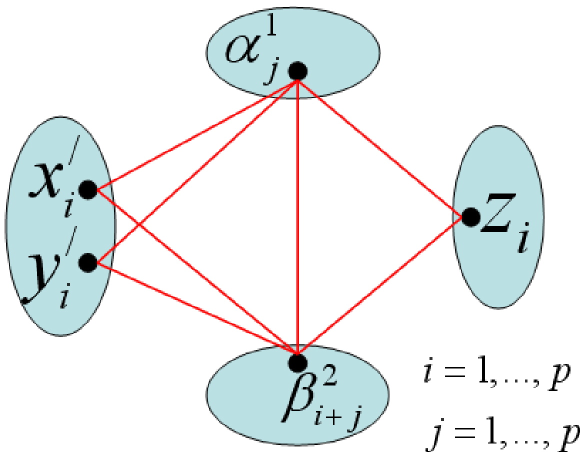


FIG. 2.18 – Graph used to match the vertices of the construction

no problems with the load constraint, again because of the sufficient condition stated in Proposition 2.1.1.

For the remaining requests, one possible idea is use the values of the grooming on the Path $(A(P_N, 2))$, which are clearly upper bounds of the values on the Ring. The following Theorem of J-C.Bermond, L.Braud and D.Coudert [4] will make our work easier.

Theorem 2.5.1 **([4])** *When N is even, $A(P_N, 2) = \left\lceil \frac{N(N-1)}{3} + \left\lceil \frac{N^2}{8} \right\rceil + \frac{N}{6} \right\rceil = \frac{11N^2-4N}{24} + \epsilon_N$, where $\epsilon_N = \frac{1}{2}$ when $N \equiv 2$ or $6 \pmod{12}$, $\epsilon_N = \frac{1}{3}$ when $N \equiv 4 \pmod{12}$, $\epsilon_N = \frac{5}{6}$ when $N \equiv 10 \pmod{12}$, and $\epsilon_N = 0$ when $N \equiv 4$ or $8 \pmod 6$. Furthermore, the construction contains $\left\lceil \frac{N^2}{8} \right\rceil$ subgraphs.*

Let's use these values of the path to cover the requests between the remaining 5 couples (realize that in this case we are also matching all the elements of each member of the

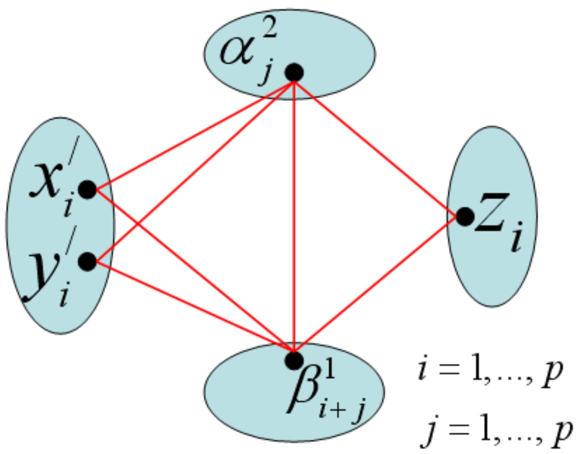


FIG. 2.19 – Graph used to match the vertices of the construction

couple among them, not only one member of the couple with the other) : $\{\alpha^1, \alpha^2\}$, $\{\beta^1, \beta^2\}$, $\{X, Y\}$, $\{Z, X'\}$ and $\{Y', Z'\}$. One can check that all the requests are already covered.
Note by $A_2(2, N)$ the number of ADMs of this construction. Taking into account all the previous calculations, we can compute :

$$A_2(2, 10p) = 20p^2 + 9p^2 + 5A(P_{2p}, 2) = 29p^2 + 5\left(\frac{11(2p)^2 - 8p}{24} + \epsilon_{2p}\right) =$$

$$= 29p^2 + \frac{5}{24}(44p^2 - 8p) + 5\epsilon_{2p} = \frac{229}{6}p^2 - \frac{p}{3} + 5\epsilon_{2p}$$

Thus, we can compute the approximation ratio of this construction :

$$\frac{A_2(2, 10p)}{Lower\ Bound} = \frac{\frac{229}{6}p^2 - \frac{p}{3} + 5\epsilon_{2p}}{\frac{11}{32}(10p)(10p-1)} \xrightarrow{\text{p}\to\infty} \frac{229}{6}\frac{32}{11}\frac{1}{100} = 1,1103$$

Note that this value is even greater than the approximation ratio that we have obtained with the generic construction of the previous section for all the odd values of N, but we will improve this construction in the next sections.

2.5.2.3 $1,0861$-approximation

It is an improvement of the previous construction. As before, $N = 10p$, and the configuration is like Figure 2.15, but now suppose in addition that $p \equiv 0 \pmod{4}$. As we have said before, most of the lack of optimality has been given by using the values of the Path. In order to minimize this effect, let's reduce the size of the sets where the optimality is lost. In Figure 2.21 we show how to math the couple $\{X, Y\}$ using $\frac{3}{4}p^2$ ADMs. Let's do the same with the other couples $\{\alpha^1, \alpha^2\}$, $\{\beta^1, \beta^2\}$, $\{Z, X'\}$ and $\{Y', Z'\}$. Taking into account the 5 couples we use $\frac{15}{4}p^2$ ADMs.
Now, we only have to match the 10 sets of p elements (instead of 5 sets of $2p$ elements

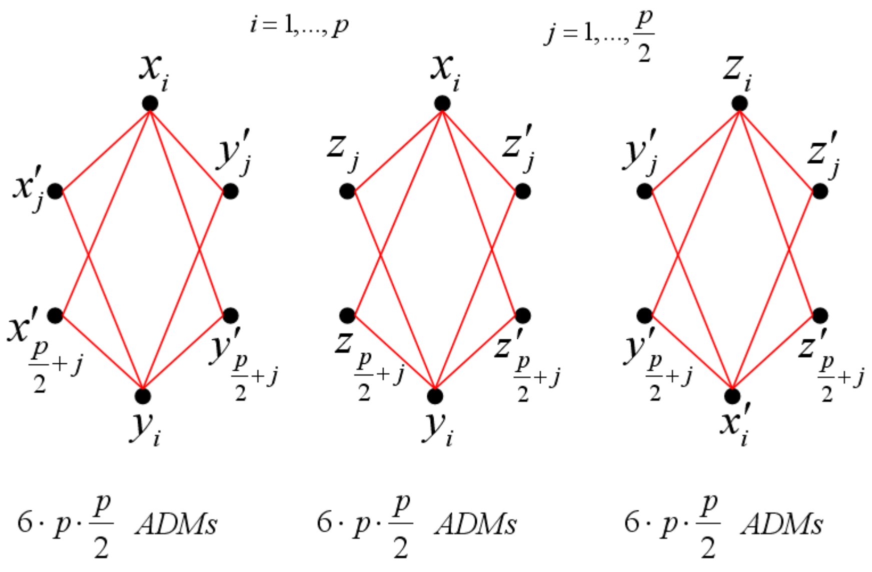


FIG. 2.20 – Graphs used to match some of the vertices of the subsets

in the previous construction). As before, note by $A_3(2,N)$ the number of ADMs of this construction.

$$A_3(2,N) = 20p^2 + 9p^2 + \frac{15}{4}p^2 + 10A(P_p,2) =$$

$$= \frac{131}{4}p^2 + 10\left(\frac{11p^2-4p}{24} + \epsilon_p\right) = \frac{448p^2-20p}{12} + 10\epsilon_p$$

Hence,

$$\frac{A_3(2,10p)}{Lower\ Bound} = \frac{\frac{448p^2-20p}{12} + 10\epsilon_p}{\frac{11}{32}(10p)(10p-1)} \overset{p\to\infty}{\longrightarrow} \frac{448}{12}\frac{32}{11}\frac{1}{100} = 1,0861$$

This time we have improved the first approximation ratio $\frac{12}{11} = 1,0909$, but on the other hand we have restricted a lot the value of N.

2.5.2.4 $1,0858$-approximation

It is again an improvement of the previous one. Provided that $p \equiv 0 \pmod 6$ and using the graph of Figure 2.22 and the Path for sets of sizes $\frac{p}{2}$ and $\frac{p}{3}$, we find a ratio of $1,0858$ (a bit lower than the previous ones). We think that it is not worth to get into details in this case due to the increasing complexity and the similarity to the other constructions.

2.5.2.5 Special decomposition for N=14

When $N=14$ there exists a kind decomposition of K_{14} into graphs with 5 vertices. Let the nodes be numbered from 0 to 13, and remark that we have to cover all the requests of length 1 until $\left\lfloor\frac{N}{2}\right\rfloor = 7$. We can use the graph of Figure 2.23, and turn it around for $i = 0\dots13$, covering in this way all the requests of length 1 to 7, because this graph has

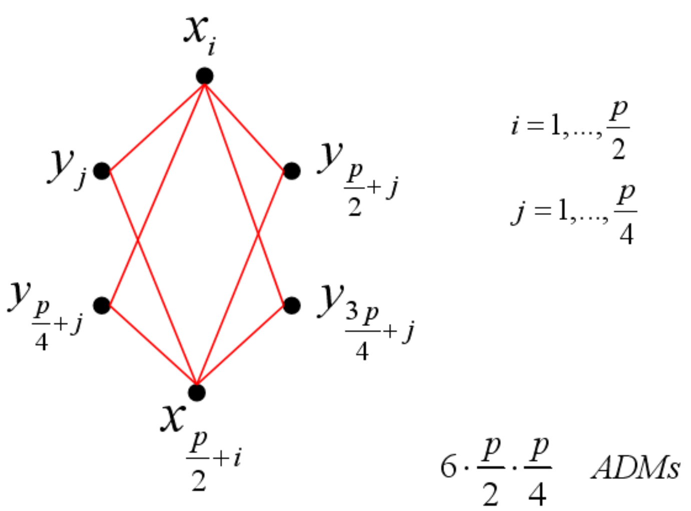


FIG. 2.21 – Matching the couple $\{X, Y\}$

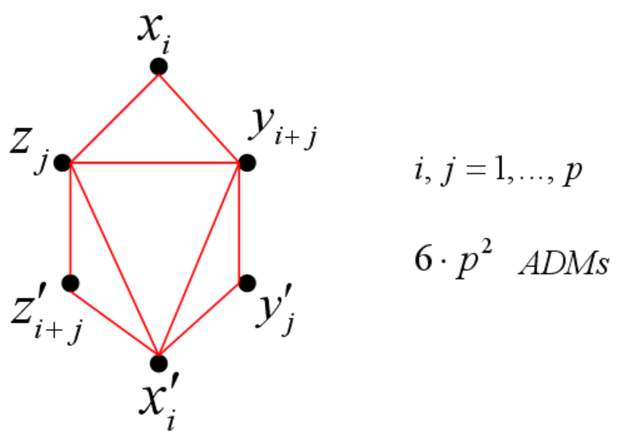


FIG. 2.22 – Matching some couples

the nice property that each of its 7 arcs has a different length (and all the lengths over the cycle add 14, luckily). I.e., for $i = 0 \dots 13$ we use the vertices

$$\{i, i+1, i+3, i+8, i+8, i+10\}$$

and the requests

$$(i, i+1),\ (i, i+3),\ (i+1, i+3),\ (i+3, i+8),\ (i+3, i+10),\ (i+8, i),\ (i+10, i)$$

So we use $14 \cdot 5 = 70$ ADMs for $N = 14$.

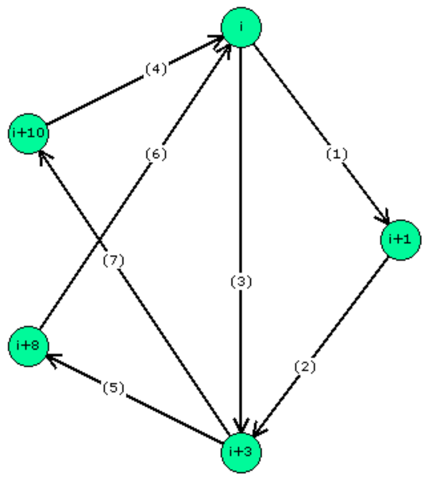


FIG. 2.23 – Graph used to partition the K_{14}

2.6 Case C=3

Although we will study later (in Section 2.7) the case $C = \frac{k(k+1)}{2}$, we will improve in this section the general results related to this value of the maximum load.

For $C = 3$ we have that $\gamma(3,2) = 1$; $\gamma(3,3) = 3$; $\gamma(3,4) = 6$ and $\gamma(3,p) = 2p$.

Equation (2.6) becomes $\sum_{p=2}^{N} a_p\gamma(3,p) = a_2 + 3a_3 + 6a_4 + \sum_{p=5}^{N} 2pa_p$.

Therefore $2A = \sum_{p=2}^{N} 2pa_p = \sum_{p=2}^{N} a_p\gamma(3,p) + 3a_2 + 3a_3 + 2a_4 = \frac{N(N-1)}{2} + 3a_2 + 3a_3 + 2a_4$.

The lower bound $\frac{N(N-1)}{4}$ can be attained if we can find a decomposition without K_2, K_3, K_4, for example with K_5 or with a graph on 6 vertices and 12 edges and supposing that the edges have length at most q.

2.6.1 Considerations about the degree

Let's do now some considerations about the best degree that we can use in the subgraphs.

Lemma 2.6.1 *For any C, if we want maximum load in all the arcs of a certain graph, then the degree of all the vertices must be even.*

Proof: At any vertex i, let´s suppose that the arc at its left in the ring has maximum load (i.e. C), and let k be the number of request that end at this vertex i. Now in the arc at its right we have load $C - k$, so if we want maximum load k requests must begin at this node i, having in this way degree $2k$, which is even. $\square$

Lemma 2.6.2 *For C=3, if we want to attain the lower bound $\frac{N(N-1)}{4}$ of Theorem 2.2.1, all the vertices must have degree 4 in all the subgraphs where they appear.*

Proof: As we have seen in Lemma 2.6.1 the best we can do is having even degree in all the vertices. Since $C = 3$, the degree can be either 2, 4 or 6. As we can see in the Figure 2.25, if one vertex has degree 6, then there are 2 vertices with degree at most 2. Taking into account these three vertices, since $4+4+4 > 6+2+2$, we conclude that it is better to use degree 4 in all the vertices (whenever is possible, of course).
Now let's check that in this way we attain the lower bound. Exactly, since in the K_N each node has degree $N-1$, each vertex will appear in $\frac{N-1}{4}$ subgraphs. Therefore, since there are N vertices, we have $\frac{N(N-1)}{4}$ ADMs, as we wanted to see. □

We can see in Table 2.4 the number of ADMs given by the Simplex and the Lower Bound for $C = 3$ and different values of $N \equiv 1 \pmod 4$.

A graph of the ratio versus $\log(N)$ is drawn in Figure 2.24 for $C = 3$ and $N \equiv 1 \pmod 4$. We can observe in this graph that the ratio begins to be equal to 1 for small values of N, and then grows a little, but insignificantly.

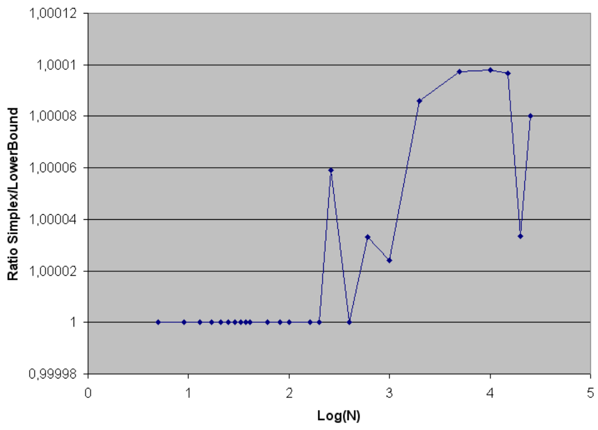


FIG. 2.24 – $\frac{ADM\ by\ Simplex}{Lower\ Bound}$ versus $\log(N)$ for $C = 3$ and $N \equiv 1 \pmod 4$

2.6.2 Tighter Lower Bounds for N=4t+3

Proposition 2.6.1 (Tighter Lower Bound for C=3 and N=4t+3) *The number of ADMs required in a bidirectional ring with N nodes and grooming factor 3, with $N \equiv 3 \pmod 4$ is lower bounded by the expression*

$$A(3, N = 4t+3) \geq \left\lceil \frac{N(3N-1)}{12} \right\rceil \tag{2.17}$$

N	$ADMs\ by\ Simplex$	$LowerBound: \left\lceil \frac{N(N-1)}{4} \right\rceil$	Ratio $\frac{ADM\ Simplex}{Lower\ Bound}$
5	5	5	1
9	18	18	1
13	39	39	1
17	68	68	1
21	105	105	1
25	150	150	1
29	203	203	1
33	264	264	1
37	333	333	1
41	410	410	1
61	915	915	1
81	1620	1620	1
101	2525	2525	1
161	6440	6440	1
201	10050	10050	1
261	16966	16965	$1,000058945$
401	40100	40100	1
601	90153	90150	$1,000033278$
1001	250256	250250	$1,000023976$
2001	1000586	1000500	$1,000085957$
5001	6251858	6251250	$1,000097261$
10001	25004947	25002500	$1,00009787$
15001	56259196	56253750	$1,000096811$
20001	100008335	100005000	$1,000033348$
25001	156268748	156256250	$1,000079984$

TAB. 2.4 – Different values of the simulations compared with the Lower Bound for $C = 3$ and $N \equiv 1 \pmod 4$

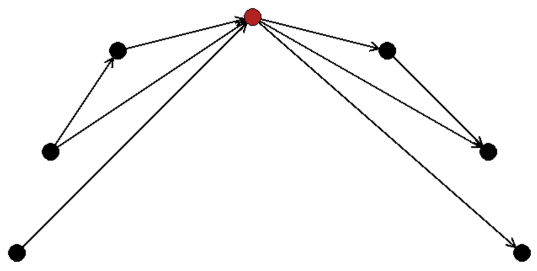


FIG. 2.25 – With $C = 3$, if one vertex has degree 6 (the red one in the figure), then there are 2 vertices (at both sides of it) with degree at most 2

Proof: The idea lies in thinking on the optimum degree. As we have seen in Lemma 2.6.2, the best we can do is to use degree 4 whenever is possible. Let δ note degree of any vertex in the complete graph. We know that $\delta = N - 1$. Since N=4t+3, $\delta \not\equiv 0 \pmod 4$. Therefore, we cannot use degree 4 in all the subgraphs, unfortunately. For each vertex, the best we can do is to use degree 4 in all the subgraphs except in one of them, where it will appear with degree either 2 or 6. As we have seen in Lemma 2.6.2, there are 2 vertices of degree 2 for each vertex of degree 6. In this way, if one vertex uses degree 6, it will appear in $\frac{N-3}{4}$ subgraphs, and if it uses degree 2 ($\frac{2}{3}$ of the vertices), it will appear in $\frac{N-3}{4} + 1$ subgraphs. Therefore now we can ensure that

$$A(3, N = 4t + 3) \geq N\frac{N-3}{4} + \frac{2}{3}N = \frac{N(3N-1)}{12},$$

as we wanted to see. □

Note that $\frac{N(3N-1)}{12} > \frac{N(N-1)}{4}$, that was the lower bound that we have got from Theorem 2.2.1.

We can see in Table 2.5 the number of ADMs given by the Simplex and the Lower Bound for $C = 3$ and different values of $N \equiv 3 \pmod 4$.

A graph of the ratio versus $\log(N)$ is drawn in Figure 2.26 for $C = 3$ and $N \equiv 3 \pmod 4$. We can see in this graph that the ratio is smaller than 1, which is an strange phenomenon. The reason is that the considerations that we have done about the degree are not taken into account in the Simplex simulations. On the other hand, the ratio tends to 1 as before.

<u>Final remark</u> : if we find a good decomposition we have also to check that the lengths of the edges are at most q. It is not necessary the case example decomposition of K_9 into C_4 $i, i+1, i+5, i+3, i$. We will discuss this later (Section 2.7).

N	$ADMs\ by\ Simplex$	$LowerBound : \left\lceil \frac{N(3N-1)}{12} \right\rceil$	Ratio $\frac{ADM\ Simplex}{Lower\ Bound}$
7	11	12	$0,916666667$
11	28	30	$0,933333333$
15	53	55	$0,963636364$
19	86	89	$0,966292135$
23	127	131	$0,969465649$
27	176	180	$0,977777778$
31	233	238	$0,978991597$
35	298	304	$0,980263158$
39	371	377	$0,984084881$
43	452	459	$0,984749455$
63	977	987	$0,989868288$
83	1702	1716	$0,991841492$
103	2627	2644	$0,993570348$
163	6602	6628	$0,996077248$
203	10252	10286	$0,996694536$
263	17228	17271	$0,997510277$
403	40502	40569	$0,998348493$
603	90753	90852	$0,998910316$
1003	251263	251419	$0,999379522$
2003	1002551	1002835	$0,999716803$
5003	6256768	6257085	$0,999949337$
10003	25014977	25014169	$1,000032302$
15003	56272017	56271252	$1,000013595$
20003	100034986	100028336	$1,000066481$
25003	156289584	156285419	$1,00002665$

TAB. 2.5 – Different values of the simulations compared with the Lower Bound for $C = 3$ and $N \equiv 3 \pmod 4$

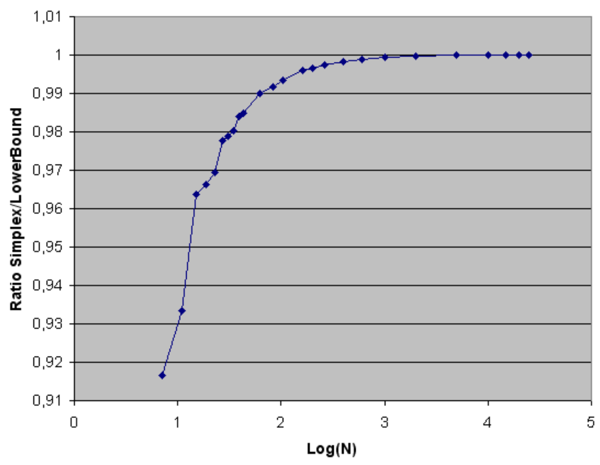


FIG. 2.26 – $\frac{ADM\ by\ Simplex}{Lower\ Bound}$ versus $\log(N)$ for $C = 3$ and $N \equiv 3 \pmod 4$

2.6.3 Optimal Decompositions

Proposition 2.6.2 *If $C = 3$, and $N \equiv 1, 5 \pmod{12}$,*

$$A(3, N) = \frac{N(N-1)}{4},$$

and therefore the General Lower Bound of Theorem 2.2.1 is achieved.

Proof: The idea is to use either the Bose or the Skolem construction to build a decomposition into triangles (see for instance [39]), and then transform this decomposition to a valid decomposition in the bidirectional case. Proposition 2.6.2 in Section 2.7 deals with a more general case, but we shall give the details here.

Consider an integer v for which we know that a Steiner triple system exists. Thus, $v \equiv 1, 3 \pmod 6$ [39]. Since each triple can be thought as a triangle, we can translate this to a decomposition of K_v into triangles. Now, label the elements as $\{\infty, 0, 1, \dots, v-2\}$. Split all the vertices i except the ∞ in two vertices i_A, i_B. We have $N = 2n - 1$, hence $N \equiv 1, 5 \pmod{12}$. Then, transform the triangles in the following form :

- If the triangle is of the form (∞, i, j), transform it into a K_5 on vertices $(\infty, i_A, j_A, i_B, j_B)$, as we can see in Figure 2.27.
- If the triangle is of the form (i, j, k), transform it into a $K_{2,2,2}$ on vertices $(i_A, j_A, k_A, i_B, j_B, k_B)$, as we can see in Figure 2.28.

We have seen that we use either K_5 (from the triangles (∞, i, j)) or $K_{2,2,2}$ (from the triangles (i, j, k)). We know that these graphs are optimal, because of Proposition 2.2.2 (or also because all the graphs used have $\rho = \rho_{min}(C)$). By definition of a Steiner triple, all the requests that we use in the decomposition are different, so we have only to check that we have the right number of requests.
Since $N = 2v - 1$, $|E| = \frac{N(N-1)}{2} = (2v-1)(v-1)$.

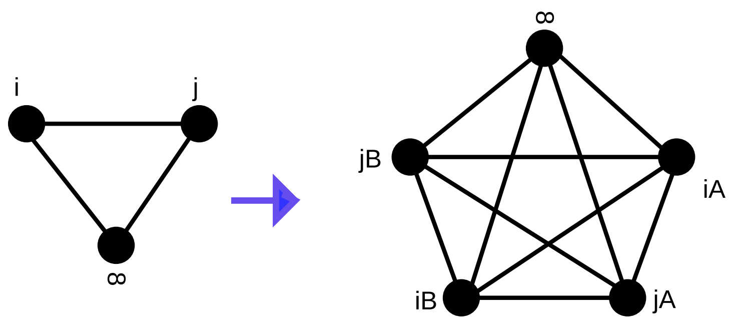


FIG. 2.27 – How to split a triangle with ∞ when $C = 3$

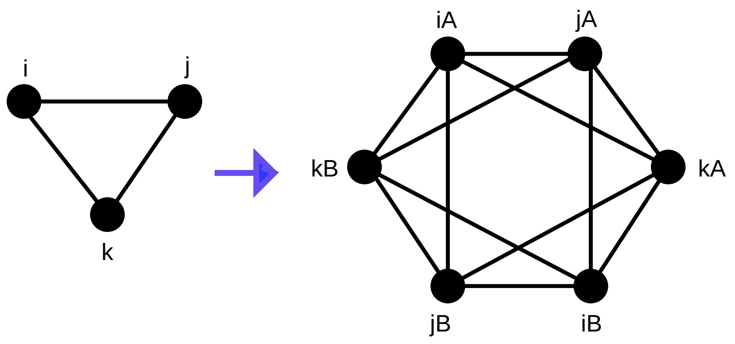


FIG. 2.28 – How to split a triangle without ∞ when $C = 3$

On the other hand, the number of Steiner triples (see [39], p.3) is $\frac{v(v-1)}{6}$. Of these triples, we have to distinguish two cases :

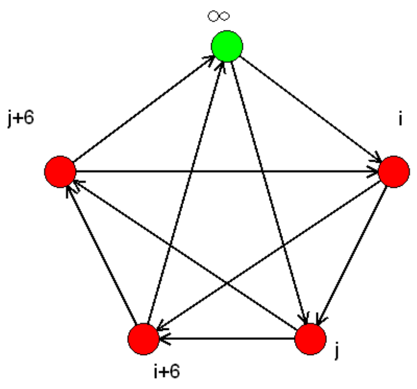
- $\frac{v-1}{2}$ triples contain the ∞, and therefore we have $\frac{v-1}{2}$ K_5 with 10 requests each graph, obtaining in this way $5(v-1)$ requests.
- All the others triples don't contain the ∞, therefore we have $\left(\frac{v(v-1)}{6} - \frac{v-1}{2}\right)$ $K_{2,2,2}$ with 12 requests each graph, obtaining $2v(v-1) - 6(v-1)$ requests.

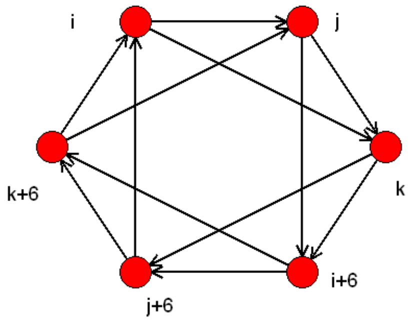
Adding up both types of requests, we obtain $2v(v-1) - v - 1 = (2v-1)(v-1)$, as we wanted to see.
Counting the number of ADMs, gives

$$5 \cdot \frac{v-1}{2} + 6\left(\frac{v(v-1)}{6} - \frac{v-1}{2}\right)$$

i.e., half the number of requests, this is, $\frac{N(N-1)}{4}$, which is the value of the lower bound. □

For example, let $N = 13$ and let the vertex set of K_{13} be $\{\infty, i, i+6\}$, $i = 0, \ldots, 5$. Then, K_{13} can be partitioned into K_5 and $K_{2,2,2}$:





$$\text{We use} \begin{cases} 3\ K_5 :\ \{\infty, i, i+3, i+6, i+9\},\ \textit{with } i = 0,1,2 \\ 4\ K_{2,2,2} :\ \{i, j, k, i+6, j+6, k+6\},\ \textit{with} \\ \qquad (i,j,k) = (0,1,5), (0,2,4), (1,2,3), (3,4,5) \end{cases}$$

Lower Bound	This construction	$\longrightarrow$ **optimal**
$\frac{N(N-1)}{4} = 39$ ADMs	$3 \cdot 5 + 4 \cdot 6 = 39$ ADMs	**decomposition !**

2.6.4 Upper Bounds and constructions

Proposition 2.6.3 *If $C = 3$, and $N \equiv 9$ (mod 12),*

$$A(3, N) \leq \frac{N(N-1)}{4} + 4$$

Proof: We have that $N = 12t+9$. Consider n such that $N = 2n-1$, and thus $n = 6t+5$. Let's consider the construction of *almost* Steiner triples for $n = 6t+5$. In this case we have all the vertices grouped into triples except one group of 5 vertices. One can check that the number of blocks of size 3 in the construction is $6t + 3\binom{2t+1}{2} = 6t^2 + 9t$. Now mark as ∞ one of the points that appear in the block of size 5. Split the blocks in the following way :

- Transform the triangles (∞, i, j) into a K_5 on vertices $(\infty, i_A, j_A, i_B, j_B)$
- Transform the triangles (i, j, k) with $i, j, k \neq \infty$ into $K_{2,2,2}$ on vertices $(i_A, j_A, k_A, i_B, j_B, k_B)$
- Transform the block of five elements (∞, i, j, k, l) into a K_9 on vertices $(\infty, i_A, j_A, k_A, l_A, i_B, j_B, k_B, l_B)$. Recall that a K_9 it is not a valid graph when the grooming factor is 3.

Let's count the number of vertices of the construction. Since the point ∞ appears in $\frac{n-1}{2} - 2$ triples, we have on this hand $5\left(\frac{n-1}{2} - 2\right)$ vertices. On the other, we have $6t^2 + 9t - \frac{n-1}{2} + 2$ triples without ∞, and thus $6\left(6t^2 + 9t - \frac{n-1}{2} + 2\right)$ vertices.
It only remains to partition the K_9 using the minimum number of vertices. Consider the K_5's $(\infty, i_A, k_A, i_B, k_B)$ and $(\infty, j_A, l_A, j_B, l_B)$, the C_4 (k_A, l_A, k_B, l_B) and a graph on

vertices $(i_A, j_A, k_A, l_A, i_B, j_B, k_B, l_B)$ with the remaining edges (one can check that it has load 3 on all the arcs). Thus, we have used $5+5+4+8=22$ ADMs.
Adding up the total number of vertices, we obtain

$$5\left(\frac{n-1}{2}-2\right)+6\left(6t^2+9t-\frac{n-1}{2}+2\right)+22=\frac{N(N-1)}{4}+4,$$

as claimed. □

Proposition 2.6.4 *If $C=3$, and $N \equiv 3,7 \pmod{12}$,*

$$A(3,N) \leq \frac{N(3N-1)}{12}+\frac{N}{3}-3$$

Proof: Let $N=2n+1$, and therefore $n \equiv 1,3 \pmod 6$)). Consider a Steiner triple system for n. We split the vertices of the triangles as before, but now we split the node ∞ into 3 nodes ∞_A, ∞_B and ∞_C.
This time, transform the triple (∞, i, j) into the K_5 on vertices $(\infty_A, i_A, j_A, i_B, j_B)$ and the graph of Figure 2.29 (i.e., using 11 vertices). Note that we can change the role of ∞_A, ∞_B and ∞_C in different graphs, and therefore we cover the 3 requests among them without using new graphs.
It is easy to check that there are $\frac{n(n-1)}{6}$ triples. Since $\frac{n-1}{2}$ of the triples contain ∞, the number of ADMs used by this construction is

$$11\frac{n-1}{2}+6\left(\frac{n(n-1)}{6}-\frac{n-1}{2}\right)=\frac{N(N-\frac{1}{3})}{4}+\frac{N}{3}-3=\frac{N(3N-1)}{12}+\frac{N}{3}-3,$$

as claimed. □

Proposition 2.6.5 *If $C=3$, and $N \equiv 11$ (mod 12),*

$$A(3,N) \leq \frac{N(3N-1)}{12}+\frac{N}{3}-11$$

Proof: It consists in combining both previous constructions. Split the node ∞ into 3 nodes ∞_A, ∞_B and ∞_C, and therefore $N=2n+1$ with $n=6t+5$. Consider an *almost* Steiner triple system for n. Adding up the ADMs coming from triples with ∞, the ADMs coming from triples without ∞ and the ADMs used in the decomposition of K_9, we obtain

$$11\left(\frac{n-1}{2}-2\right)+6(6t^2+6t)+22=\frac{N(3N-1)}{12}+\frac{N}{3}-11,$$

as claimed. □

Note that we have already studied all the cases for $C=3$ and N odd, obtaining optimal decompositions for $N \equiv 1,5 \pmod{12}$, and 1-approximations for all the other cases.

We sum up the main results of this section in Table 2.6.

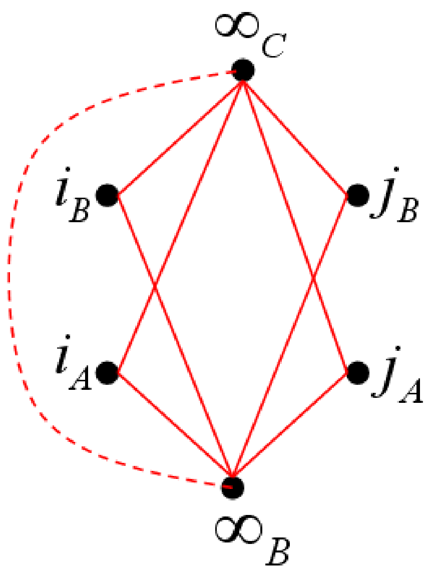


FIG. 2.29 – Graph used in the construction when $N \equiv 3,7 \pmod{12}$

2.7 C of the form $\sum_{i=1}^{k} i$

2.7.1 Optimal Solution using BIBDs

In this section we will extend the optimal construction that we have found for $C = 3$ in Proposition 2.6.2. Again, we will split the vertices and here the basic tool from design theory will be a BIBD.

Our aim is to find the minimum number of ADMs of the bidirectional case. We can summarize our method as follows :

1. We consider C of the form $C = \sum_{i=1}^{k} i = \frac{k(k+1)}{2}$
   For example, $C = 1, 3, 6, 10...$

2. We want to find a decomposition in graphs that achieve the $\rho_{min}(C)$. For each C, consider a generic number of vertices v. Let´s make a partition of vertices into an isolate vertex (let's call it ∞ as it is usual in this kind of constructions) and sets of $2k$ (remember $k = k(C)$) vertices surrounding the central vertex. Obviously, it will not be possible for a generic value of N.

3. Match the vertices of the sets of $2k$ vertices and identify both vertices of the pair. Doing this we obtain a new graph G on N vertices (this graphs have a special vertex labeled as ∞ and sets of k vertices). It's easy to check that $N = 2(v-1)+1 = 2v-1$.

4. Now we use the optimal results of the unidirectional case to make an optimal decomposition of the new graph G. To be able to make the optimal decomposition, it will give us congruences for N modulo P for a certain P, and therefore congruences for v modulo P.

$N \pmod{12}$	Lower Bound	ADMs of the explicit constr.	Ratio $\frac{construction}{lower\ bound}$
1	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4}$	1
3	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 3$	$1 + \frac{\frac{N}{3}-3}{\frac{N(3N-1)}{12}} \overset{N\to\infty}{\longrightarrow} 1$
5	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4}$	1
7	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 3$	$1 + \frac{\frac{N}{3}-3}{\frac{N(3N-1)}{12}} \overset{N\to\infty}{\longrightarrow} 1$
9	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4} + 4$	$1 + \frac{4}{\frac{N(N-1)}{4}} \overset{N\to\infty}{\longrightarrow} 1$
11	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 11$	$1 + \frac{\frac{N}{3}-11}{\frac{N(3N-1)}{12}} \overset{N\to\infty}{\longrightarrow} 1$

TAB. 2.6 – Explicit constructions for C=3 and N odd

5. The idea is to convert each subgraph of the unidirectional case to a subgraph of the bidirectional case without losing the optimality. The subgraphs of the unidirectional case are K_{k+1}. Remember that each vertex of the graph G (except the center) corresponds to a pair of vertices of the real bidirectional graph. We consider two cases :

   - If the K_{k+1} contains the center, we transform it into a complete graph K_{2k+1}
   - If the K_{k+1} doesn't contain the center, we transform it into a complete multipartite graph $K_{2\times(k+1)}$

   We have already seen examples about how to transform a triangle ($k = 2$, $C = 3$) without the vertex ∞ in Figure 2.28, and with it in Figure 2.27.

   We have seen in Section 2.1.3 that the labeling of the vertices is a key issue. Nevertheless, for this specific construction we can give a labeling to the vertices that ensures the fulfilment of the load constraint. Remember that, if $N = 2p + 1$, we have two sets of p vertices, and an isolated vertex. Let's label as ∞ this isolated vertex, and label as $\{i_A, i_B\}$ each pair of split vertices, $0 \le i \le p-1$. Thus, the vertex set is $\{\infty, 0_A, 1_A, \ldots, (p-1)_A, 0_B, 1_B, \ldots, (p-1)_B\}$, and in this way there are no requests of length greater than $p = \left\lfloor \frac{N}{2} \right\rfloor$ (it is not difficult to check).

6. As we know (see previous section) that this subgraphs achieve the ρ_{min}, we have found an optimal decomposition with the minimum number of ADMs. As we have said, the well known results for the unidirectional case give us congruences about N. To find the congruences for n we only have to use that $N = 2n - 1$.

7. Using this construction and the results taken from [22] that we have extended using [15], we obtain Table 2.7. What we have to do is just look for a BIBD(v, k, 1), with $k = 1, 2, 3 \ldots$. Then we have to transform the values of v that we obtain to $N = 2v - 1$. The problem is that nowadays only BIBDs of small block size are known. Furthermore, for $C = 15, 21, 28, 37$ there is a finite set of values of N for

which the existence of a BIBD that guarantees an optimal construction remains unknown [15]. We can see an exhaustive list of these values in Table 2.8. For all the other values of N, the rule of Table 2.7 applies.

k	C	$\bar{\delta}$	ρ_{min}	N
1	1	2	1	$N \equiv 1 \pmod{2}$
2	3	4	1/2	$N \equiv 1,5 \pmod{12}$
3	6	6	1/3	$N \equiv 1,7 \pmod{24}$
4	10	8	1/4	$N \equiv 1,9 \pmod{40}$
5	15	10	1/5	$N \equiv 1,9 \pmod{30}$
6	21	12	1/6	$N \equiv 1,13 \pmod{84}$
7	28	14	1/7	$N \equiv 1,15 \pmod{112}$
8	37	16	1/8	$N \equiv 1,17 \pmod{144}$

TAB. 2.7 – Congruences for N and certain values of the grooming factor C

It is easy to check that we really have a construction with grooming factor $C = \frac{k(k+1)}{2}$. Indeed, either in a K_{2k+1} or in a $K_{2\times(k+1)}$, each vertex is joint with its k nearest neighbours in each direction of the cycle (remember that we are always "inside" a ring), i.e. we have all the requests of lengths 1 till k, and thus a load of $\frac{k(k+1)}{2}$, as we wanted to see.

In this way we have found an optimal decomposition of the complete graph K_N.

Theorem 2.7.1 *For the values of C and N found via the previous process, the lower bound of Theorem 2.2.1 is attained, and therefore we have found an optimal decomposition.*

Proof: It is straightforward from the previous explanation, and taking into account the optimality of the graphs used in the decomposition (Proposition 2.2.2). □

C	*Possible exceptions of optimal constructions*
15	91 101 121 161 281 331 391 401 451 461 511 521 551 571 581 631 641 691 701 751 811 821 871 881 941 991 1001 1051 1121 1181 123 1291 1301 1351 1411 1481 1531 1541 1591 1601 1831 1891 2441 2491 2501 2791 2911 2971 3041 3221 3341 3701 4061
21	169 253 265 349 421 505 517 1009 1177 1429 1597 1609 1849 2269 2437 2449 2773 2869 3109 3277 3289 3529 3613 3709 3949 4957 5125 5209 5545 5629 6973
28	225 337 351 449 561 673 785 1247 1471 1569 2129 2241 2255 2353 2465 2479 2591 2689 2801 2815 2913 2927 3025 3039 3137 3151 3473 3585 3809 3921 4369 4481 5153 5825 6609 6833 6945 7505
37	289 305 433 449 577 593 721 737 1009 1153 1585 1729 1745 1889 2033 2161 2609 2881 2897 3025 3169 3185 3329 3457 3617 3761 3889 3905 4049 4465 4481 4609 4769 4897 4913 5329 5473 5489 5761 5777 5921 6049 6193 6209 6481 6641 6769 6785 6929 7201 7489 7505 7633 8065 8513 8641 8657 8785 8801 8929 8945 9649 9665 9793 9809 10369 10801 10945 10961 12097 12257 13249 13393 13409 13537 13553 13825 13969 14689 14705 14849 15409 18433 19009 20033 21329 25057 25073 26369 27505 27665 27937 28225 28369 28945 29105 29249 29377 30097 30113 30529 31537 32993

TAB. 2.8 – Finite sets of values of N for which the existence of a BIBD remains undecided

2.8 Large values of C

We have that, in the all-to-all case, the load of each arc is $\frac{N^2-1}{8}$ if $N = 2q+1$, and $\frac{N}{4}(\frac{N}{2}+1)$ if $N = 2q$ (this is in the worst case, where all the q diameters cross that edge). In any case we can write it as $\sum_{i=1}^{q} i = \frac{q(q+1)}{2}$ to simplify.
If $C \geq \frac{N^2-1}{8}$ then obviously we will put all the requests in only one subgraph, using in this way N ADMs.
Now let´s see what happens for specific values of C :

- Let $N = 2q$, and q even. Let´s put all the requests of half of the nodes ($\frac{N}{2}$) in one subgraph. We have to compute now what is the load that we have in this case. It's not difficult to see that in such a situation there are the same number of requests of odd length than in the all-to-all case, and half of the requests of even case.

Lemma 2.8.1 *Let $N = 2q$, and q even. When $C = \frac{q}{4}(\frac{3}{2}q+1)$, we can use $\frac{3}{2}N$ ADMs.*

Proof: It´s enough to see that if we put all the requests of half of the nodes (taken from 2 to 2 in the whole graph) in one subgraph, the load is exactly that grooming factor. Then, we have also to check that we can satisfy the remaining requests of the other half of the nodes in other subgraph respecting the load.
First of all, observe that in a given arc, we will have all the requests of odd length, because those requests will always contain a node of the half that we want to cover all the requests. On the other hand, half of the requests of even side will begin and end in the other half of the nodes, and so they will not be taken into account in this subgraph, having in these way only half of the requests of even length. Therefore, in conclusion, the load of this subgraph can we computed as :

$$load = 1+3+5+...+(q-1)+1+2+3+...+\frac{q}{2} = \sum_{i=1}^{\frac{q}{2}} i + \sum_{i=1}^{\frac{q}{2}} (2i-1) = ... = \frac{q}{4}(\frac{3}{2}q+1)$$

Now we have to cover all the requests among the $\frac{N}{2}$ remaining nodes in another subgraph. Now the load is $\sum_{i=1}^{\frac{q}{2}} i = \frac{q}{4}(\frac{q}{2}+1)$, that is clearly lower than the grooming factor.
In conclusion, we have used $N + \frac{N}{2} = \frac{3}{2}N$ ADMs, as we wanted to see. □

- Let $N = 2q$, and q odd. We follow the same process than in the previous case.

Lemma 2.8.2 *Let $N = 2q$, and q odd. When $C = \frac{q+1}{4}(\frac{3}{2}q+\frac{1}{2})$, we can use $\frac{3}{2}N$ ADMs.*

Proof: Now the new load is :

$$load = 1+3+5+...+q+1+2+3+...+\frac{q-1}{2} = \sum_{i=1}^{\frac{q-1}{2}} i + \sum_{i=1}^{\frac{q+1}{2}} (2i-1) = ... = \frac{q+1}{4}(\frac{3}{2}q+\frac{1}{2})$$

The rest of the proof is the same. □

Note that in both cases the grooming factor required is larger than half of the total load ($\frac{q}{4}(q+1)$), which is a fact that can maybe contradict intuition.

2.9 MILP formulation of the Problem using CPLEX

A MILP formulation is useful to guess if the lower bounds that we have are very far away from the real ones.

For example, for $C = 2$, we have seen that the simulated values of the number of ADMs tend to the value that we state in Proposition 2.5.1 as N increases.

In Appendix A we can see the C code compiled to formulate the problem in terms of Linear Programming and an example of the output file of the CPLEX software.

2.10 Concluding Remarks

In this Chapter we have studied the Bidirectional Ring Grooming Problem with all-to-all unitary requests and shortest path symmetric routing. We have found a new lower bound, after finding the explicit formula of $\gamma(C,p)$, which was an open problem. We have improved this lower bound for $C = 2, 3$. Some upper bounds have also been provided.

We have focused mainly on the cases $C = 2$ and $C = 3$, finding in both cases either optimal solutions or α-approximations, being α very close to 1. By using the existence of a BIBD, we have found optimal constructions for these values of C and N :

C	N
1	$N \equiv 1 \pmod{2}$
3	$N \equiv 1, 5 \pmod{12}$
6	$N \equiv 1, 7 \pmod{24}$
10	$N \equiv 1, 9 \pmod{40}$
15	$N \equiv 1, 9 \pmod{30}$
21	$N \equiv 1, 13 \pmod{84}$
28	$N \equiv 1, 15 \pmod{112}$
37	$N \equiv 1, 17 \pmod{144}$

A MILP formulation of the problem has been done in order to get an idea about the tightness of our lower bound. Finally, some results have been found for large values of C.

Chapter 3

Other Sets of Requests

Abstract

In this Chapter study other particular cases of the general Traffic Grooming Problem. First of all, we consider the unidirectional ring with a set of requests made up of graphs with a bounded degree.
In the second part we focus on the bidirectional ring case, with circulant graphs as set of requests.

3.1 Degree constraints in the unidirectional case

An interesting grooming problem arises from considering graphs of requests in which we have restrictions on the degree of the vertices, namely we are given a maximum degree. Here we study the unidirectional ring (using *primitive rings*, each pair of requests is represented by an edge, following the standard notation of [22]) and the particular case when we have constant degree in all the vertices. The idea lies in thinking about the "worst case" set of requests, because an important feature of the problem is that we have to place the ADMs in the vertices *before* knowing which is the set of requests. We solve here the cases corresponding to degree 2 and 3.

Note by $ADM(\Delta = \delta = k, N, C)$ the optimal solution in a unidirectional ring of N nodes with grooming factor C when the set of requests is given by a k-regular graph.

3.1.1 2-regular graphs of requests

We consider degree equal to 2 in all the vertices (2-regular graphs of requests). In this case the set of requests is made up by disjoint cycles. The idea is to consider all the possible request graphs. Then, besides the small values of N, we have that

Proposition 3.1.1 *If the set of requests is given by a 2-regular graph, then*

$$ADM(\Delta = \delta = 2, N, C) = 2N - (C - 1)$$

Proof: Since the degree is 2, of course a possible solution consists in placing 2 ADMs at each vertex. What we do is to count in how many ADMs we can assure that we can place only an ADM. As we have said, we have to think in the worst case.
Let's see first that we cannot use 1 ADM in more than $C - 1$ vertices. Suppose this, i.e. that we have 1 ADM in C vertices and 2 in all the others. Then, consider a set of requests made up by a cycle of length $C + 1$ that has all C vertices with 1 ADM inside it. In this situation, we are forced to use 2 subgraphs, and at least 2 vertices must appear in both subgraphs, hence we will need more than 1 ADM in some vertex that had initially only 1 ADM.
Now, let's see that we can always save $C - 1$ ADMs. Let $\{a_0, a_1, \ldots, a_{C-2}\}$ be the set of vertices with only 1 ADM. We will see that we can decompose the set of requests in such a way that the vertices a_i always lie in the middle of a path or a cycle, covering in

this way both requests of each vertex with only 1 ADM. Indeed, if two of these vertices (namely, a_i and a_j) are not consecutive in one of the disjoint cycles of the set of requests. Let b_i be the nearest vertex to a_i in the cycle in the direction of a_j, and conversely for b_j (b_i may be equal to b_j if a_i and a_j differ only on one vertex). Then, consider two paths (or cycles) of the form $\{b_i a_i \ldots\}$ and $\{b_j a_j \ldots\}$ to assure that both a_i and a_j lie in the middle. We can see an illustration in Figure 3.1
Otherwise, suppose that all $C-1$ vertices are consecutive in a cycle. Let b_0 be the nearest vertex to a_0 different from a_1, and let b_{C-2} be the nearest vertex to a_{C-2} different from a_{C-3}. Then, consider a subgraph with the path (or cycle, if $b_0 = b_{C-2}$) $\{b_0 a_0 a_1 \ldots a_{C-2} b_{C-2}\}$. □

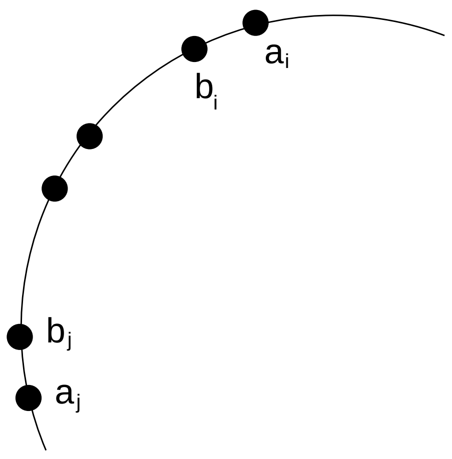


FIG. 3.1 – Construction in the proof of Proposition 3.1.1

3.1.2 3-regular graphs of requests

In this case we will need some previous graph theory concepts.

Definition 3.1.1 (Separate) *If $A, B \subseteq V$ and $X \subseteq V \cup E$ are such that every* A-B *path in G contains a vertex or an edge from X, we say that X* separates *the sets A and B in G. More generally we say that X* separates *G if $G - X$ is disconnected, that is, if X separates in G some two vertices that are not in X. A separating set of vertices is a* separator.

Definition 3.1.2 (Cutvertex ; Bridge) *A vertex which separates two other vertices of the same component is a* cutvertex*, and an edge separating its ends is a* bridge*. Thus, the bridges in a graph are precisely those edges that do not lie on any cycle.*

We can see a graph with a bridge in 3.2. The ends of the bridge are cutvertices.

It is time to recall some other definitions given in Chapter 1.

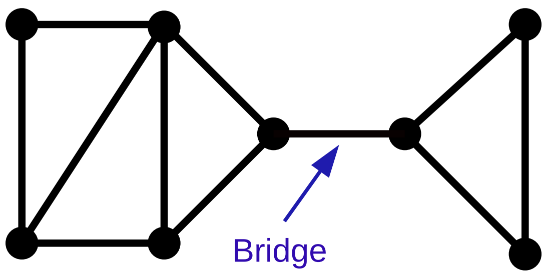


FIG. 3.2 – Example of a graph with a bridge

Definition 3.1.3 (Matching) *A set M of independent edges in a graph $G = (V, E)$ is called a* matching*. M is a matching of $U \subseteq V$ if every vertex in U is incident with an edge in M. The vertices in U are then called* matched *(by M); vertices not incident with any edge of M are* unmatched*.*

Definition 3.1.4 (k-factor) *A k-regular spanning subgraph is called a k-factor. Thus, a subgraph $H \subseteq G$ is a 1-factor of G if and only if $E(H)$ is a matching of V.*

We use now a well known result from matching theory :

Theorem 3.1.1 (Petersen, 1981) *Every bridgeless cubic graph has a 1-factor.*

Then, if we erase a 1-factor from a cubic graph, what it remains is a disjoint set of cycles.

Corollary 3.1.1 *Every bridgeless cubic graph has a decomposition into a 1-factor and disjoint cycles.*

We can see an example of a decomposition of a bridgeless cubic graph into disjoints cycles and a 1-factor in Figure 3.3.

We will use Proposition 3.1.1 in the following result.

Proposition 3.1.2 *If the set of requests is given by a bridgeless cubic graph and $C = 3$, then*

$$ADM(\Delta = \delta = 3, N, 3) = 2N$$

Proof: When $C = 3$ and we cannot assure the existence of triangles, the best we can do is to decompose the set of requests into paths of length 3. Let's proof that we can do it. In the decomposition of Proposition 3.1.1, take a clockwise orientation of the edges of each cycle. With this orientation, each edge of the 1-factor has two "incoming" and two "outgoing" edges of the cycles. Now take for each edge of the 1-factor the 2 incoming edges to it, and form in this way a path of length 3. It is easy to verify that this is a true decomposition into paths of length 3. For instance, if we do this in the graph of Figure 3.3, and we label the edges of the 1-factor as {A,B,...,G} and the ones of the cycles as {1,2,...,14} (see Figure 3.4), we have the following decomposition of the graph :

$$\{1, A, 6\}, \{5, B, 2\}, \{3, C, 8\}, \{7, D, 9\}, \{14, E, 11\}, \{10, F, 12\}, \{4, G, 13\}$$

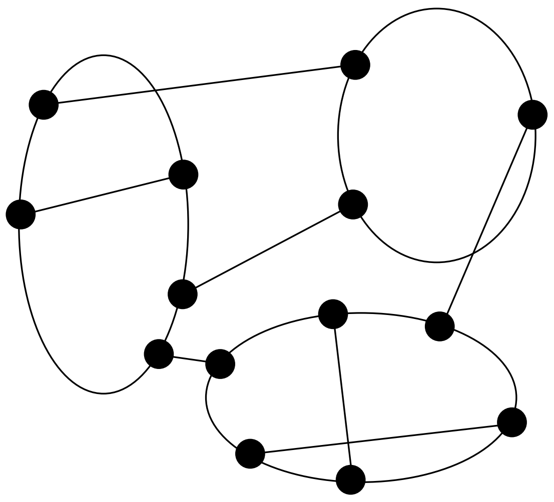


FIG. 3.3 – Decomposition of a bridgeless cubic graph into disjoints cycles and a 1-factor

Now let's see that we cannot do it better, that is, with $2N - 1$ ADMs. If such a solution exists, there would be at least one vertex with only 1 ADM, and the mean of the ADMs of all other vertices must not exceed 2. In order to see that this is not always possible, consider the cubic bridgeless graph of Figure 3.5. Let C be the vertex with only 1 ADM. This graph has the property that is has no triangles except of those through C. Since we can use only 1 ADM in C, we must take all its requests in one subgraph (green color). Now, it is not possible to cover the 4 remaining requests of the nodes A and B in one single subgraph (the best we can do is to take the red path), and thus wlog we will need 3 ADMs in A. With these constraints, one can check that the best solution we can find uses 20 ADMs, that equals $2N > 2N - 1$. □

Looking at the proof, we see that the only thing that we need from the bridgeless cubic graph is that it has a decomposition into a 1-factor and disjoint cycles. Hence, we can relax a bit the hypothesis of Proposition 3.1.2.

Corollary 3.1.2 *If $C = 3$, and the set of requests can be partitioned into disjoints cycles and a 1-factor, then*

$$ADM(\Delta = \delta = 3, N, 3) = 2N$$

Another corollary can be deduced easily.

Corollary 3.1.3 *If $C \geq 3$, and the set of requests can be partitioned into disjoints cycles and a 1-factor, then*

$$ADM(\Delta = \delta = 3, N, C) \leq 2N$$

Proof: We can do the same construction of Proposition 3.1.2 to obtain $2N$ ADMs. □

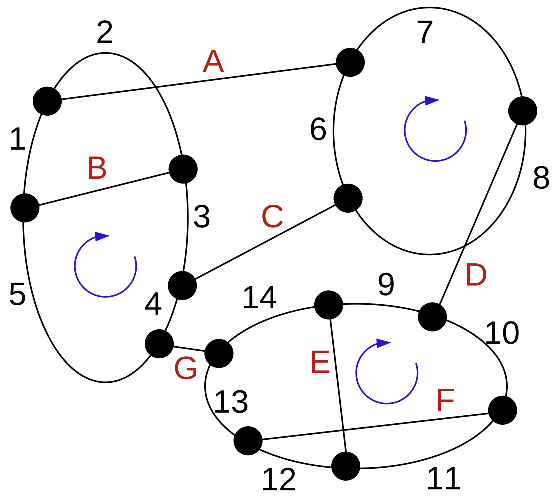


FIG. 3.4 – Decomposition of a bridgeless cubic graph into paths of length 3

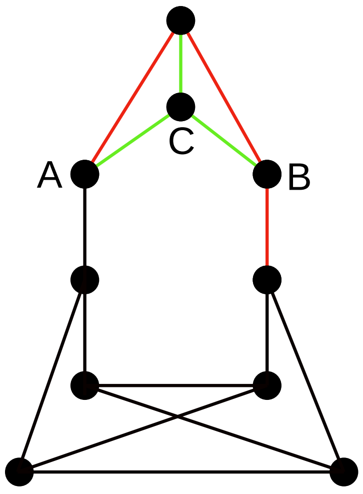


FIG. 3.5 – Cubic bridgeless graph used in the proof of Proposition 3.1.2

3.1.3 *k*-regular graphs of requests

The main fact that obstructs the number of ADMs to fall down is that we cannot assure the existence of cycles of length smaller or equal than C. In this case, the best graphs we can use are paths.

Proposition 3.1.3 *If the set of requests is given by a k-regular graph, then*

$$ADM(\Delta = \delta = k, N, C) \geq \frac{Nk}{2C}(C + 1)$$

Proof: The proof consists just in counting how many ADMs would be in a decomposition of the request set into stars with C edges. Since the number of edges is $\frac{Nk}{2}$, the number of stars is $\frac{Nk}{2C}$, and this yields the result taking into account that each star has $C+1$ vertices. $\square$

A natural question is : when can we assure the existence of cycles of length smaller or equal than C ?

Let I be the request graph, as usual. If $N \geq E(I)$ then we know that there are cycles (because I is not a tree in this case), but these cycles can be very large, until length N. Thus, a sufficient condition is that $C \geq N \geq E(I)$.

3.2 Circulant graphs in the bidirectional case

Definition 3.2.1 *A* circulant graph *is a graph of n vertices in which the ith vertex is adjacent to the $(i+j)$th and $(i-j)$th vertices for each j in a list l. The circulant graph $Ci_n(1, 2, \ldots, \lfloor \frac{n}{2} \rfloor)$ gives the complete graph K_n and the graph $Ci_n(1)$ gives the cyclic graph C_n.*

The number of circulant graphs on $n = 1, 2, \ldots$ nodes (counting empty graphs as circulant graphs) are $1, 2, 2, 4, 3, 8, 4, 12, \ldots$, the first few of which are illustrated in Figure 3.6. Note that these numbers cannot be counted simply by enumerating the number of nonempty subsets of $\{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}$ since, for example, $Ci_5(1) = Ci_5(2) = C_5$. There is an easy formula for prime orders, and formulas are known for square-free and prime-squared orders.

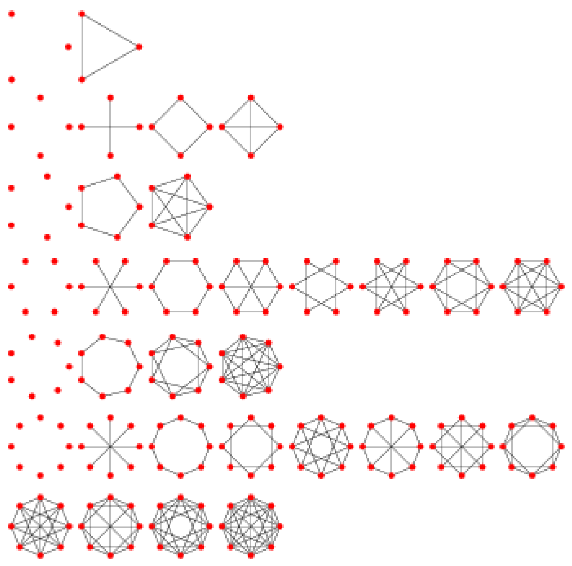


FIG. 3.6 – Circulant graphs enumeration

The numbers of connected circulant graphs on $n = 1, 2, \ldots$ nodes are $0, 1, 1, 2, 2, 5, 3, 8, \ldots$, illustrated in Figure 3.7.

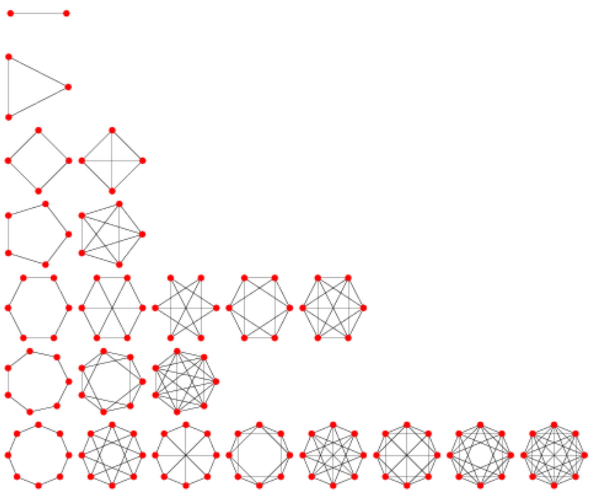


FIG. 3.7 – Connected circulant graphs enumeration

We will use the funny fact that the numbers from 1 till 12 can be packed into 4 sets of 3 elements such that in each set 2 of its elements add up the third. One possible way to do it is :

$$\begin{aligned} 1+6&=7\\ 2+10&=12\\ 3+8&=11\\ 4+5&=9 \end{aligned}$$

Note that the solution might not be unique. Indeed :

$$\begin{aligned} 3+9&=12\\ 4+7&=11\\ 2+8&=10\\ 1+5&=6 \end{aligned}$$

Proposition 3.2.1 *If $N = 39k$, $C = 2$, and the set of requests in a ring is given by a circulant graph $Ci_n(1, 2, 3, \ldots, 12)$, then we have an optimal decomposition using $8N$ ADMs.*

Proof: First of all, note that the best we can do in the ring with grooming factor 2 is use whenever possible triangles joined by a vertex. Any other configuration have smaller ratio $\frac{edges}{vertices}$, as it is not difficult to check.
If $N = 39k$, we will see now that we can decompose the whole set of requests into subgraphs made up by joined triangles. Indeed, let $N = 39$, and use the graph of Figure 3.8

turning it around for $i = 0, \dots, 38$. In this way, we have covered all the requests of lengths $1, \dots, 12$, and we have no repeated requests. With this construction we use $8 \cdot 38 = 8N$ ADMs.
Now let $N = 39k$, with $k > 1$. What we do now is to merge k of the previous graphs in the following way : when the $4th$ triangle of the $1st$ graph, begin with the $1st$ triangle of the $2nd$, and so on. We obtain a graph on $8k$ vertices. We can see in Figure 3.9 an example for $k = 2$, i.e $N = 78$.
Then, turn this big graph around for $i = 0, \dots, 38$. As before, we have covered all the requests of lengths $1, \dots, 12$, and we have no repeated requests. With this construction we use $8k \cdot 38 = 8N$ ADMs, as claimed. □

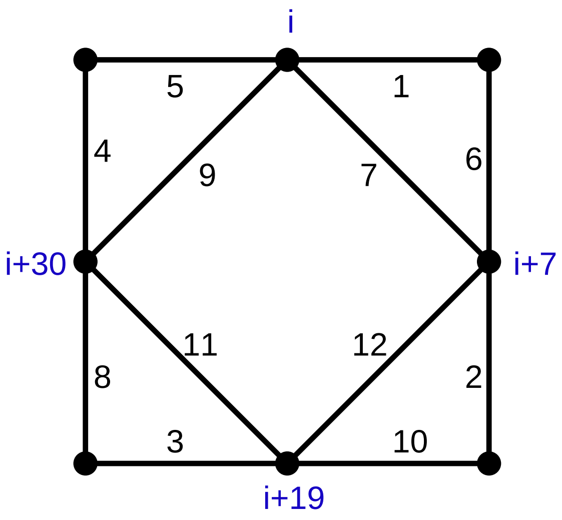


Fig. 3.8 – Special decomposition of a circulant graph

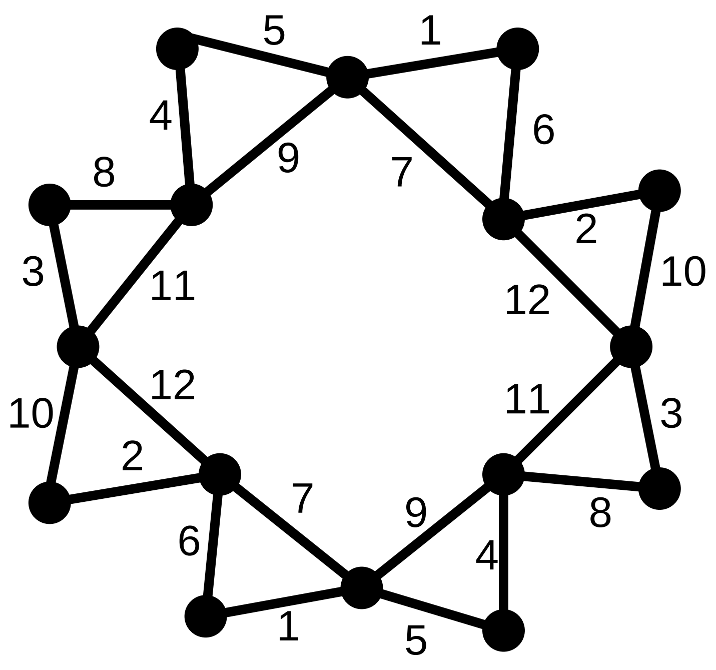


FIG. 3.9 – Merging 2 optimal graphs for $N = 78$

3.3 Concluding Remarks

We have begun this Chapter considering the unidirectional ring with a set of requests made up of graphs with a bounded degree. We have solved the cases when the graph is 2-regular and 3-regular. Lower bounds have been deduced in the k-regular case.

In the second part we have dealt with the bidirectional ring case, with circulant graphs as set of requests. We have found an optimal solution for a particular case.

Chapter 4

Grooming for Two-Period Optical Networks

Abstract

In this Chapter study a variation of the Traffic Grooming Problem : the Grooming for Two-Period Optical Networks. We consider two possible values of the grooming factor, that affect different subsets of the node set of the network.
It is a first approach to the dynamic grooming using graph partitioning tools.

4.1 Introduction

The study of this problem has been motivated by an article named *Grooming for Two-Period Optical Networks* that has been written by Charles J. Colbourn, Gaetano Quattrocchi and Violet R. Syrotiuk, and that has not been published yet [17].

As we have already said before, many approaches to the *traffic grooming* problem are possible. Even restricting to the ring topology, also quite different scenarios have been considered, with respect to different features of the network :

- All-to-all, one-to-all or irregular traffic pattern
- Uniform or non-uniform traffic. In the non-uniform case, for instance one can consider a distance-dependent traffic (the amount of traffic between node pairs is inversely proportional to the distance separating them) or a hub traffic (all the traffic is going to one node on the ring)
- Bidirectional or unidirectional routing
- Static or dynamic traffic

Heuristics and approximations have been widely applied for nearly all scenarios, but conversely graph theoretical tools have been applied above all for the static traffic case. Only in [12] they apply some tools from graph theory for the dynamic grooming problem. The main idea in that paper by Colbourn et al. is extending the theoretical methodology stated in [10] to some kind of dynamic traffic pattern.

Specifically, they consider the problem of *minimizing the drop cost to support two traffic periods in SONET/WDM unidirectional rings.* Informally, each time period supports different traffic requirements. In the first time period n nodes are required to support a grooming ratio of C, while in the second time period a grooming ratio of C', $C' < C$, is required for $v \leq n$ nodes. We can think in applying a static theoretical approach for each time period. This allows the two-period grooming problem to be expressed as an optimization problem on graph decompositions of K_n that embed graph decompositions of K_v for $v \leq n$. Using this formulation, optimal two-period groomings are found for small grooming ratios. In the next section we will formalize the problem and the notation, following the cited article.

4.2 Definitions and Statement of the Problem

In general, the traffic grooming problem consists in partitioning the set of traffic requirements into a number of groups such that each group is carried on a single SONET/WDM ring. When the traffic requirements are uniform symmetric with grooming ratio C and the ring has n vertices, the 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.

Let's make now the notion of a two-period optical network precise.

In a network with n nodes, the *traffic requirement* R_{ij} between the node pair $i \leftrightarrow j$ is represented by the $n \times n$ array $\mathcal{R} = [R_{ij}]$. If $R_{ij} = \frac{1}{C} > 0$ for every pair of distinct nodes i, j, then the traffic requirements of the network are uniform symmetric with grooming ratio C.

Definition 4.2.1 *Let $X = \{0, 1, \ldots, n-1\}$ and $V \subseteq X$, with $|V| = v$. A* grooming of a two-period network *$N(n, v; C, C')$ with ratio (C, C') is a SONET/WDM network and a partition of time into two disjoint periods T and T' such that :*

1. *For times in T the traffic is symmetric and uniform with ratio $C \geq 2$ between all node pairs in V; that is $R_{ij}^{T} = \frac{1}{C}$, $C \geq 2$, for every $i, j \in V$, $i \neq j$.*
2. *For times in T' the traffic is symmetric and uniform with ratio C', $1 \leq C' < C$, between all node pairs in V, i.e., $R_{ij}^{T'} = \frac{1}{C'}$, $C \geq 2$, for every $i, j \in V$, $i \neq j$. For every pair $\{i, j\}$ such that either $i, j \in W = X \setminus V$ or $i \in X$ and $j \in W$, $R_{ij}^{T'} = 0$.*

Intuitively, the traffic in time periods T and T' is specified by different requirements matrices R_{ij}^{T} and $R_{ij}^{T'}$. In the first time period, the traffic between node pairs of the n nodes has a grooming ratio of C. In the second time period only a subset of $v \leq n$ nodes are active with the traffic among them therefore able to run at a higher rate (and hence $C' < C$).

Let's model the problem of grooming in a two-period network in terms of graph decomposition. Recall that a graph decomposition of the complete graph K_n on n vertices is a partition of the edges of K_n into b edge-disjoint subgraphs G_i, $i = 0, 1, \ldots, b-1$. Such a decomposition is denoted by $(V, \mathcal{B})$, where V is the vertex set of K_n and $\mathcal{B} = \{G_0, G_1, \ldots, G_{b-1}\}$. Following terminology from design theory, $\mathcal{B}$ is the *block set* of the graph decomposition.

Definition 4.2.2 *Let $V \subseteq X$, $V \neq \emptyset$. 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 (that is, one-to-one), then $(X, \mathcal{B})$* faithfully embeds *$(V, \mathcal{D})$.*

A grooming of a two-period network $N(n, v; C, C')$ with ratio (C, C') coincides with a graph decomposition $(X, \mathcal{B})$ of K_n such that $(X, \mathcal{B})$ is a grooming $N(n, C)$ in time period T, and $(X, \mathcal{B})$ faithfully embeds a graph decomposition of K_v such that $(V, \mathcal{D})$ is a grooming $N(v, C')$ in time period T'. Furthermore, for every $B \in \mathcal{B}$ such that $f^{-1}(B) \neq \emptyset$, $\sum_{D \in f^{-1}(B)} |E(D)| \leq C'$.

Simply put, a $N(n, v; C, C')$ coincides with an $N(n, C)$ that faithfully embeds an $N(v, C')$.

Let's use the same notation that Colbourn et al. for referring to the standard graphs :

- $K_{V,W}$ denotes a complete bipartite graph in which the classes are V and W.
- $P_n = [a_0, a_1, \ldots, a_{n-1}]$ denotes a path, i.e., $V(P_n) = \{a_0, a_1, \ldots, a_{n-1}\}$ and $E(P_n) = \{a_0a_1, a_1a_2, \ldots, a_{n-2}a_{n-1}\}$.
- $K_n = (a_0, a_1, \ldots, a_{n-1})$ denotes a complete graph, i.e., $V(K_n) = \{a_0, a_1, \ldots, a_{n-1}\}$ and $E(K_n) = \{a_0a_1, a_0a_2, \ldots, a_0a_{n-1}, a_1a_2, a_1a_3, \ldots, a_1a_{n-1}, \ldots, a_{n-2}a_{n-1}\}$. If the notation K_V is used, it represents the complete graph on the vertex set V.
- $K_{1,n} = [a_0; a_1, a_2, \ldots, a_n]$ is used for the star where a_0 is the source (center) and $a_1, \ldots, a_n$ are sinks, i.e., $V(K_{1,n}) = \{a_0, a_1, \ldots, a_n\}$ and $E(K_{1,n}) =$ $= \{a_0a_1, a_0a_2, \ldots, a_0a_n\}$.

We use the notation $\mathscr{ON}(n, v; C, C')$ to denote an optimal grooming $N(n, v; C, C')$ of a two-period SONET/WDM network with ratio (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* $\mathscr{ON}(n, v; C, C') >$ *cost* $\mathscr{ON}(n, C)$. Of particular interest is the case when *cost* $\mathscr{ON}(n, v; C, C') =$ *cost* $\mathscr{ON}(n, C)$.

For every triple (n, C, C') denote by $\aleph(n, C, C')$ the set of integers v for which *cost* $\mathscr{ON}(n, v; C, C') =$ *cost* $\mathscr{ON}(n, C)$. In this definition it is implied that in each case we choose the best set V to achieve $\mathscr{ON}(n, v; C, C')$, because in general *cost* $\mathscr{ON}(n, v; C, C')$ will depend on V. Evidently $1 \in \aleph(n, C, C')$ for every positive integer n.

In the article the authors have provided complete solutions when $(C, C') =$ (2,1), (3,1) and (3,2), respectively. Here we will extend the results by studying the case $(C, C') = (4, 1)$, using similar techniques as Colbourn et al.

4.3 $\mathscr{ON}$(n, v ; 4, 1)

We recall the complete characterization for optimal groomings with a grooming ratio of four :

Theorem 4.3.1 ([10]) *If $n \geq 6$, an $\mathscr{ON}(n,\, 4)$ can be realized with*

- $\left\lceil \frac{n(n-1)}{8} \right\rceil - \alpha$ *C_4's and $K_3 + e$;*
- *α K_3,*

where

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

Furthermore, the number of subgraphs is the minimum : $\left\lceil \frac{n(n-1)}{8} \right\rceil$

We can see the graphs involved in this decomposition in Figure 4.1.

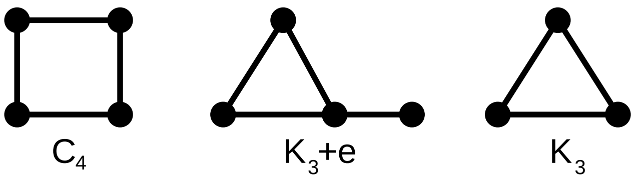


FIG. 4.1 – Graphs involved in the decomposition of K_n when $C = 4$

In this case, there will be two different problems :

- Variant A : we just want an optimal $\mathscr{ON}(n, C)$, that is, we allow as many K_3s as we want.
- Variant B : we are looking for an optimal $\mathscr{ON}(n, C)$ with the minimum number of wavelengths. Thus, there are at most 3 K_3s.

We will see that the results for both variants of the problem are different. We will specify in each results to which variant we refer.

Recall that $V = \{0, 1, \ldots, v-1\}$ and that $W = \{a_0, a_1, \ldots, a_{n-v-1}\}$. Note that for all $V \subseteq X$ we can decompose the whole graph K_n into a K_V, a K_W and a $K_{V,W}$.

Now, let

$$\vartheta_4^A(n) = \left\lceil \frac{n}{2} \right\rceil$$

and

$$\vartheta_4^B(n) = \begin{cases} 2 & \text{if } n = 3 \\ 3 & \text{if } n = 4 \\ 4 & \text{if } n = 7 \\ \lfloor \frac{n}{2} \rfloor & \text{if } n > 7 \end{cases}$$

Proposition 4.3.1 *With the definitions above,*

Variant A : $\forall v \in \aleph(n, 4, 1)$, $v \leq \vartheta_4^A(n)$.

Variant B : $\forall v \in \aleph(n, 4, 1)$, $v \leq \vartheta_4^B(n)$.

Proof: Among the graphs of an optimal solution (C_4, K_3 and $K_3 + e$) consider those intersecting V in at least (and therefore exactly, because $C' = 1$) one edge. They are of 3 types :

- Type 1 : those intersecting W in one edge and containing two edges between V and W. They correspond to numbers $1, 2, 5$ in Figure 4.2.
- Type 2 :those not intersecting W and containing 3 edges between V and W. They correspond to numbers $3, 4$ in Figure 4.2.
- Type 3 :those not intersecting W and containing two edges between V and W. They correspond to number 6 in Figure 4.2.

Let a_1, a_2, a_3 be the number of subgraphs of the $\mathscr{ON}(n, 4)$ of type $1, 2, 3$.

Counting respectively the edges in V, those between V and W and those in W we obtain :

$$\begin{aligned} a_1 + a_2 + a_3 &= \binom{v}{2} && (4.1) \\ 2a_1 + 3a_2 + 2a_3 &\leq v(n - v) && (4.2) \\ a_1 &\leq \binom{n-v}{2} && (4.3) \end{aligned}$$

From (4.1) and (4.2) we deduce

$$0 \leq a_2 \leq v(n - v) - 2\binom{v}{2} = v(n - 2v + 1) \qquad (4.4)$$

Therefore $v \leq \lceil \frac{n}{2} \rceil$, proving the result for variant A.

If n is odd and $v = \frac{n+1}{2}$, then $a_2 = 0$, but from (4.1) and (4.3) we deduce that $a_3 \geq \binom{v}{2} - \binom{n-v}{2} = \frac{n-1}{2}$.

So, as soon as $n \geq 9$ and n odd, there cannot exist an optimal solution with the minimum number of wavelengths and so in variant B : $v_4(n) \leq \lfloor \frac{n}{2} \rfloor$, proving the result for variant B. □

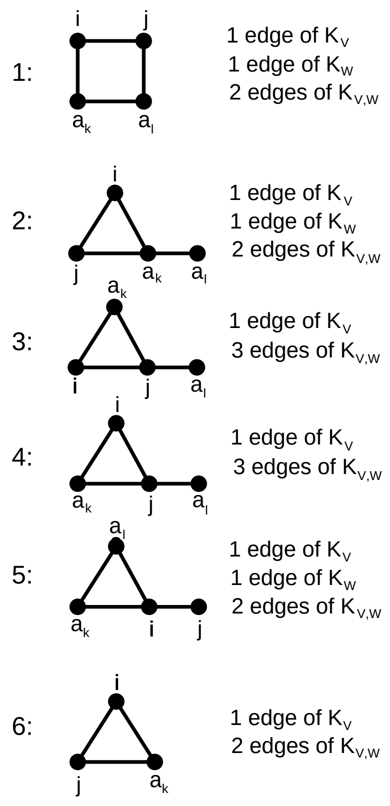


FIG. 4.2 – Possible configurations for $(C, C') = (4, 1)$

Remark 4.3.1 *If $v \in \aleph(n, C, C')$, then $w \in \aleph(n, C, C')$ for all $1 \leq w \leq v$.*

Proof: If $w < v$, then $|K_w| < |K_v|$. □

Proposition 4.3.2 *We have that*

Variant A : $\aleph(n, 4, 1) = \{1, 2, \ldots, \vartheta_4^A(n)\}$. *Equivalently, for* $v \leq \vartheta_4^A(n)$,

$$\text{cost } \mathscr{ON}(n, v; 4, 1) = \text{cost } \mathscr{ON}(n, 4) = \frac{n(n-1)}{2}$$

Variant B : $\aleph(n, 4, 1) = \{1, 2, \ldots, \vartheta_4^B(n)\}$. *Equivalently, for* $v \leq \vartheta_4^B(n)$,

$$\text{cost } \mathscr{ON}(n, v; 4, 1) = \text{cost } \mathscr{ON}(n, 4) = \frac{n(n-1)}{2}$$

Proof: [Proof for variant A]

Because of Remark 4.3.1, it will suffice to prove that $\vartheta_4^A(n) \in \aleph(n, 4, 1)\ \forall\ n$.

Case n = 3, v= 2 : 1 K_3 OK.

Case n odd : $n = 2t + 1$, $v = t + 1$:

Proof by induction that there exists an optimal $\mathscr{ON}(2t+1, t+1; 4, 1)$ with t K_3 and $\binom{t}{2}$ C_4's. Suppose it is true for $n = 2t+1$ and $v = t+1$; add two vertices ∞ and a_∞, the C_4's $(i, \infty, a_i, a_\infty)$ $i = 0, 1, \dots, t-1$ and the K_3 (t, ∞, a_∞).

Remark : Note that for $n = 3, 5, 7$ the solution has the minimum number of K_3 and so the minimum number of wavelengths.

Case n even : $n = 2t$, $v = t$:

Proof by induction that there exists an $\mathscr{ON}(2t, t; 4, 1)$ with at least one C_4 containing an edge of V.
Example for $n = 8$: decomposition into 6 $K_3 + e$'s and 1 C_4 :
$(1, 2, a_3)+(a_0, a_3)$; $(0, 3, a_2)+(a_1, a_2)$; $(1, 3, a_1)+(3, a_0)$; $(1, a_0, a_2)+(0, 1)$; $(2, a_0, a_1)+(0, 2)$; $(0, a_1, a_3) + (0, a_0)$; and the C_4 $(2, 3, a_3, a_2)$.

Suppose it is true for $n = 2t$ and $v = t$ and let the C_4 be $(t-2, t-1, a_{t-1}, a_{t-2})$; add two vertices ∞ and a_∞ ; then delete the C_4 $(t-2, t-1, a_{t-1}, a_{t-2})$, add the $K_3 + e$: $(t-2, a_{t-2}, \infty) + (\infty, a_\infty)$, the three K_3's $(t-1, a_{t-1}, \infty)$, $(a_{t-2}, a_{t-1}, a_\infty)$ and $(t-2, t-1, a_\infty)$ and the $(t-2)$ C_4's $(i, \infty, a_i, a_\infty)$ for $i = 0, 1, \dots, t-3$.

For $n = 6$ and $v = 3$ we have a decomposition with 3 $K_3 + e$ and one K_3 :
$(0, 1, a_0) + (1, a_2)$; $(1, 2, a_1) + (2, a_0)$; $(0, 2, a_2) + (0, a_1)$; (a_0, a_1, a_2).

Remark : Note that for $n = 6, 8, 10$ the $\mathscr{ON}(n, 4)$ has also the minimum number of wavelengths. □

Proof: [Proof for variant B]

Because of Remark 4.3.1, it will suffice to prove that $\vartheta_4^A(n) \in \aleph(n, 4, 1)$ $\forall$ n.

Case n even : $n = 2t$, $v = t$:

Proof by induction that there exists an $\mathscr{ON}(2t, t; 4, 1)$ with minimum number of wavelengths.
Induction from t to $t + 4$, that is from $n = 2t$, $v = t$ to $n = 2t + 8$, $v = t + 4$.
It works for $n = 6, 8, 10$ (remark above). For $n = 12$ start from the solution for $n = 8$, delete the C_4 $(2, 3, a_3, a_2)$, add 4 vertices $\infty_0, \infty_1, a_{\infty_0}, a_{\infty_1}$, the $K_3 + e$'s $(\infty_0, \infty_1, a_{\infty_1})$ + $(a_{\infty_0}, a_{\infty_1})$; $(2, \infty_0, a_2) + (\infty_0, a_{\infty_0})$; $(2, 3, a_{\infty_0}) + (a_{\infty_0}, \infty_1)$ plus the K_3's $(3, \infty_0, a_3)$ and (a_2, a_3, a_{∞_0}). Then add the C_4's $(i, \infty_0, a_i, a_{\infty_0})$ for $i = 0, 1$ and $(i, \infty_1, a_i, a_{\infty_1})$ for $i = 0, 1, 2, 3$.

Suppose it is true for $n = 2t$ and $v = t$; add the 8 vertices ∞_j, a_{∞_j} for $j = 0, 1, 2, 3$. Take the decomposition for n, v and on the graph on the 8 vertices ∞_j, a_{∞_j} for $j = 0, 1, 2, 3$ plus the C_4's $(i, \infty_j, a_i, a_{\infty_j})$ for $i = 0, 1, \dots, t-1$ and $j = 0, 1, 2, 3$.

Case n odd : $n = 2t + 1$, $v = t$:

Proof by induction that there exists an $\mathscr{ON}(2t, t; 4, 1)$ with minimum number of wavelengths.

Induction from t to $t + 4$, that is from $n = 2t + 1$, $v = t$ to $n = 2t + 9$, $v = t + 4$.

True for $n = 3, 5, 7$ (see remark above).

True for $n = 9$ and $v = 4$ with 9 $K_3 + e$: $(0, 1, a_0) + (a_0, a_3)$; $(0, 2, a_1) + (a_1, a_3)$; $(0, 3, a_2) + (a_2, a_3)$; $(2, 3, a_0) + (a_0, a_4)$; $(1, 3, a_1) + (a_1, a_4)$; $(1, 2, a_3) + (3, a_3)$; $(0, a_3, a_4) + (3, a_4)$; $(1, a_2, a_4) + (2, a_4)$; $(a_0, a_1, a_2) + (2, a_2)$.

Now suppose it is true for $n = 2t + 1$ and $v = t$; add the 8 vertices ∞_j, a_{∞_j} for $j = 0, 1, 2, 3$. Take the decomposition for n,v and on the graph on the 9 vertices ∞_j, a_{∞_j} for $j = 0, 1, 2, 3$ plus the vertex a_t. Finally use the C_4's $(i, \infty_j, a_i, a_{\infty_j})$ for $i = 0, 1, \ldots, t-1$ and $j = 0, 1, 2, 3$. □

Proposition 4.3.3 (Lower Bound) *For both variants of the problem, if $v > \vartheta_4^A(n)$ (resp. $\vartheta_4^B(n)$), then*

$$cost\ \mathscr{ON}(n, v; 4, 1) \le \frac{n(n-1)}{2} + \binom{v}{2} - \left\lfloor \frac{v(n-v)}{2} \right\rfloor$$

Proof: Note that for an edge of K_v there can be two possibilities :

- Case 1 : either it belongs to a C_4, or a K_3 or a $K_3 + e$, in which case this subgraph contains 2 edges of $K_{V,W}$.
- Case 2 : it belongs to another kind of subgraph for which the number of vertices is one more than its number of edges. We can see all these graphs in Figure 4.3.

Let e_1 be the number of edges of type 1 and e_2 of type 2. We have :

$$e_1 + e_2 = \binom{v}{2} \tag{4.5}$$

$$2e_1 \le v(n-v) \tag{4.6}$$

$$A = \binom{n}{2} + e_2 \tag{4.7}$$

Therefore, if $v > \lceil \frac{n}{2} \rceil$, then $v(n-v) \ge \binom{v}{2}$ and so $e_2 \ge 0$.

From (4.6) and (4.5) we get : $e_2 \ge \binom{v}{2} - \lfloor \frac{v(n-v)}{2} \rfloor$ implying by (4.7) the lower bound for *cost* $\mathscr{ON}(n, v; 4, 1)$. □

For the next theorem the only way to obtain an optimal cost for $ON(n, v; 4, 1)$ is to allow triangles.

Theorem 4.3.2 *For variant A of the problem, if $v > \vartheta_4^A(n)$, then*

$$cost\ \mathscr{ON}(n, v; 4, 1) = \frac{n(n-1)}{2} + \binom{v}{2} - \left\lfloor \frac{v(n-v)}{2} \right\rfloor$$

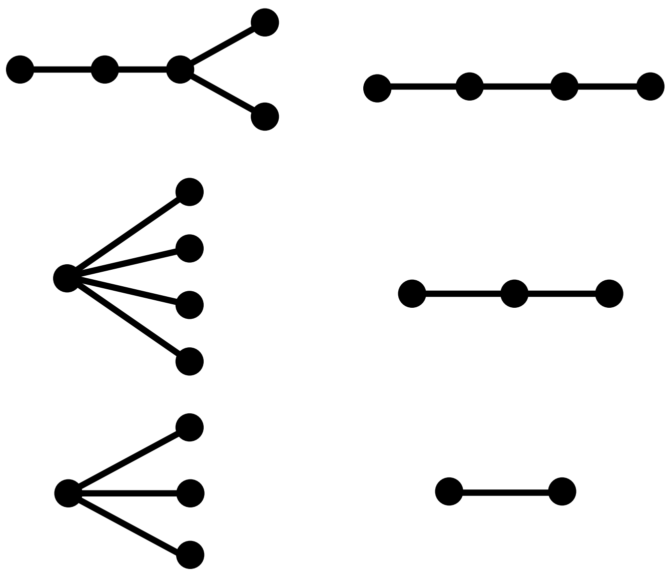


FIG. 4.3 – Other possible graphs in a $N(n, v; 4, 1)$

Proof:

Proof of equality, case v even :

Partition K_v into $v-1$ 1-factors F_i for $i = 0, 1, \dots, v-2$.
For $j = 0, 1, \dots, n-v-1$ join a_j to the edges of F_j to form $\frac{v(n-v)}{2}$ triangles.
Add to $\binom{n-v}{2}$ of these triangles the edges of K_W ; that is possible since $\binom{n-v}{2} \leq \frac{v(n-v)}{2}$. For example add to the K_3's containing a_j the edges $a_j, a_j + k$ for $k = 0, 1, \dots, \frac{n-v}{2}$ the indices being taken modulo $n-v$ and when $n-v$ is even the edges $a_j, a_{j+\frac{n-v}{2}}$ being added only once.
Finally it remains $\binom{v}{2} - \frac{v(n-v)}{2}$ edges of K_v (those of F_j for $j = n-v, \dots, v-1$) which gives the excess value to ON(n,4) in the cost formula.

Proof of equality, case v odd :

Partition K_v into v near 1-factors F_i, where i is missing in F_i, for $i = 0, 1, \dots, v-1$. (F_i consists of the edges $i-k, i+k$ for $k = 1, 2, \dots, \frac{v-1}{2}$).
For $j = 0, 1, \dots, n-v-1$ join a_j to the edges of F_j to form $\frac{v(n-v)}{2}$ triangles.
Form the $\lfloor \frac{n-v}{2} \rfloor$ C_4's $(2h, a_{2h}, a_{2h+1}, 2h+1)$ for $h = 0, 1, \dots, \lfloor \frac{n-v}{2} \rfloor - 1$.
Add to $\binom{n-v}{2} - \lfloor \frac{n-v}{2} \rfloor$ of the triangles the edges of K_W like for v even.
Finally it remains $\binom{v}{2} - \lfloor \frac{v(n-v)}{2} \rfloor$ edges of K_v which gives the excess value to *cost* $\mathscr{ON}(n, 4)$ in the cost formula. □

4.4 Concluding Remarks

In this Chapter we have studied the Grooming for Two-Period Optical Networks. This is a first theoretical approach to dynamic grooming.
Following the graph partitioning methods of [17], we have completely solved the case $(C, C') = (4, 1)$.

Chapter 5

Conclusions and Further Research

5.1 Conclusions

This Master Thesis deals with the Traffic Grooming Problem in Optical WDM Rings.

In Chapter 1, after providing all the necessary concept for understanding this work, we have formulated the Traffic Grooming problem in terms of graph partitioning, and we have shown that it turns out to be a combinatorial optimization problem.
We have also summarized the state-of-the-art, and finally we have seen that the Grooming Problem is NP-complete.

In Chapter 2 we have studied the Bidirectional Ring Grooming Problem with all-to-all unitary requests and shortest path symmetric routing. We have found a new lower bound, after finding the explicit formula of $\gamma(C, p)$, which was an open problem. We have improved this lower bound for $C = 2, 3$. Some upper bounds have also been provided.
We have focused mainly on the cases $C = 2$ and $C = 3$, finding in both cases either optimal solutions or α-approximations, being α very close to 1. By using the existence of a BIBD, we have also found optimal constructions for these values of C and N :

C	N
1	$N \equiv 1 \pmod{2}$
3	$N \equiv 1, 5 \pmod{12}$
6	$N \equiv 1, 7 \pmod{24}$
10	$N \equiv 1, 9 \pmod{40}$
15	$N \equiv 1, 9 \pmod{30}$
21	$N \equiv 1, 13 \pmod{84}$
28	$N \equiv 1, 15 \pmod{112}$
37	$N \equiv 1, 17 \pmod{144}$

A MILP formulation of the problem has been done in order to get an idea about the tightness of our lower bound. Finally, some results have been found for large values of C.

We have begun Chapter 3 considering the unidirectional ring with a set of requests made up of graphs with a bounded degree. We have solved the cases when the graph is 2-regular and 3-regular. Lower bounds have been deduced in the k-regular case.
In the second part we have dealt with the bidirectional ring case, with circulant graphs as set of requests. We have found an optimal solution for a particular case.

Finally, in Chapter 4 we have studied the Grooming for Two-Period Optical Networks. Following the methods of [17], we have solved the case $(C, C') = (4, 1)$.

5.2 Future work and open problems

A lot of exciting problems remain unsolved concerning the Traffic Grooming problem. Let's cite some of them :

- We have given a general lower bound. Is it possible to improve this lower bound for other values of C (besides 2 and 3) ?
- Does the grooming problem belong to FPT ?
- Consider the grooming problem in the Bidirectional Ring but not imposing neither shortest path nor symmetric routing.
- In the Ring and for each value of N and C, which is the routing that minimizes the number of ADMs ?
- Consider the grooming problem in a Tree.
- Consider the grooming problem in the Unidirectional Ring with constant degree (greater than 2) in the request graph.
- Try to improve the best ratio of an approximation algorithm known up to date for the traffic grooming problem.
- Try to solve more cases of the Two-Period Grooming problem.

Appendix A

MILP formulation of the problem

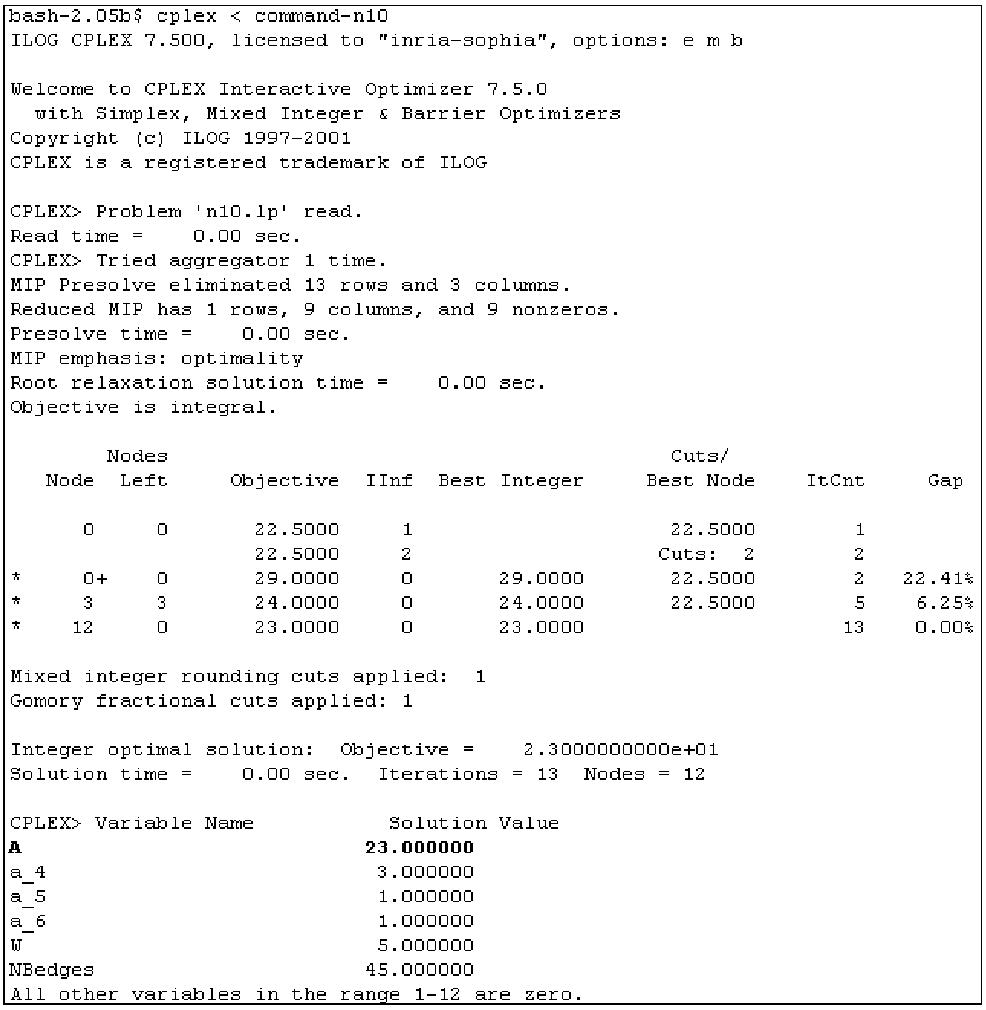
In Section A.1 we can see the code that we have compiled to simulate the problem, using the CPLEX software. In Figure A.1 in Section A.2 we can see an example of an output file of this software of Linear Programming.

A.1 C source code for MILP formulation in Chapter 2

```
#include <stdio.h>\\
#include <stdlib.h>\\
#include <math.h>\\\\


int my_gamma(int C,int N) {\\
  int p,r;\\

  if(N==2) return 1;
  if(N==3) return 3;
  if(N==4) if(C==2) return 5;

  if(C==2) return (3*N)/2;
  else
    {
      p= floor( (1+sqrt(1+8*C))/(double)2);
      r = C-p*(p-1)/2;

      return (p-1)*N+floor(r*N/p);
    }
}


void gen_lp(FILE *f,int C,int N) {
  int i;
  int W;

  fprintf(f,"\\Problem name: blop for C=%d and N=%d\n",C,N);
  fprintf(f,"Minimize\n  A\n\nSubject To\n\n");

  fprintf(f,"\\NUMBER OF ADMS\n");

  fprintf(f,"A");
  for(i=2;i<=N;i++)
    fprintf(f," - %d a_%d",i,i);

  fprintf(f," >= 0\n");
  fprintf(f,"\n\\NUMBER OF SUBGRAPHS\n");
  fprintf(f,"W");
  for(i=2;i<=N;i++)
    fprintf(f," - a_%d",i);

  fprintf(f," <= 0\n");
  fprintf(f,"W >= %d",(int)ceil((N*N-1)/(8*(double)C)));
  fprintf(f,"\n\\NUMBER OF EDGES\n");
  fprintf(f,"NBedges");
  for(i=2;i<=N;i++)
    fprintf(f," - %d a_%d",my_gamma(C,i),i);

  fprintf(f," <= 0\n");
  fprintf(f,"NBedges >= %d",N*(N-1)/2);
  fprintf(f,"\n\\VARIABLES POSITIVES\n");
  for(i=2;i<=N;i++)
    fprintf(f,"a_%d >= 0\n",i);

  fprintf(f,"\n\\VARIABLES ENTIERES\nGENERAL\n");
  for(i=2;i<=N;i++)
    fprintf(f,"a_%d\n",i);

  fprintf(f,"\nEnd\n");
}


void write_command(int C,int N) {
  FILE *f;
  char buf[500];
  sprintf(buf,"C%d/command-n%d",C,N);
  f=fopen(buf,"w");
  fprintf(f,"read n%d.lp\nmip\ndisp sol var -\n",N);
  fclose(f);
}
```

```
main(int argc, char **argv) {
  int C,N;
  int i;

  FILE *f;
  char buf[500];

  sscanf(argv[1],"%d",&C);
  sscanf(argv[2],"%d",&N);

  f=stdout;

  sprintf(buf,"C%d/n%d.lp",C,N);
  f=fopen(buf,"w");

  //fprintf(stdout,"%d\n",my_gamma(C,N));

  gen_lp(f,C,N);

  fclose(f);

  write_command(C,N);
}
```

A.2 Example of an output file of CPLEX

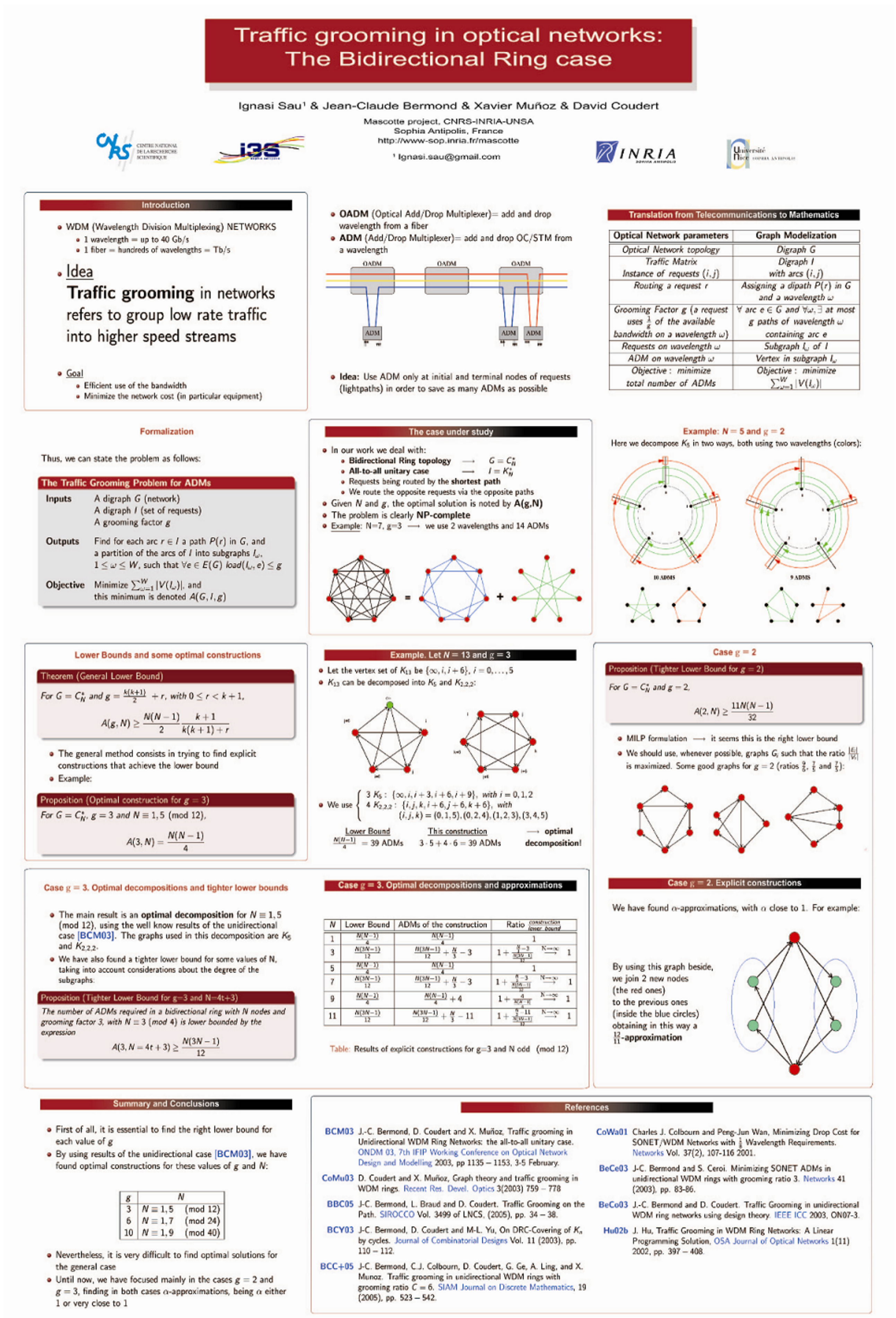


```
bash-2.05b$ cplex < command-n10
ILOG CPLEX 7.500, licensed to "inria-sophia", options: e m b

Welcome to CPLEX Interactive Optimizer 7.5.0
  with Simplex, Mixed Integer & Barrier Optimizers
Copyright (c) ILOG 1997-2001
CPLEX is a registered trademark of ILOG

CPLEX> Problem 'n10.lp' read.
Read time =    0.00 sec.
CPLEX> Tried aggregator 1 time.
MIP Presolve eliminated 13 rows and 3 columns.
Reduced MIP has 1 rows, 9 columns, and 9 nonzeros.
Presolve time =    0.00 sec.
MIP emphasis: optimality
Root relaxation solution time =    0.00 sec.
Objective is integral.

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0       22.5000     1                     22.5000        1
                    22.5000     2                    Cuts:  2        2
*     0+    0       29.0000     0       29.0000       22.5000        2   22.41%
*     3     3       24.0000     0       24.0000       22.5000        5    6.25%
*    12     0       23.0000     0       23.0000                     13    0.00%

Mixed integer rounding cuts applied:  1
Gomory fractional cuts applied: 1

Integer optimal solution:  Objective =    2.3000000000e+01
Solution time =    0.00 sec.  Iterations = 13  Nodes = 12

CPLEX> Variable Name           Solution Value
A                            23.000000
a_4                           3.000000
a_5                           1.000000
a_6                           1.000000
W                             5.000000
NBedges                      45.000000
All other variables in the range 1-12 are zero.
```

Fig. A.1 – Example of an output file of CPLEX (C=3, ADMs=23)

Appendix B

Poster in ADONET/COST293 Budapest



Traffic grooming in optical networks: The Bidirectional Ring case

Ignasi Sau[1] & Jean-Claude Bermond & Xavier Muñoz & David Coudert

Mascotte project, CNRS-INRIA-UNSA
Sophia Antipolis, France
http://www-sop.inria.fr/mascotte

[1] Ignasi.sau@gmail.com

Introduction

- WDM (Wavelength Division Multiplexing) NETWORKS
  - 1 wavelength = up to 40 Gb/s
  - 1 fiber = hundreds of wavelengths = Tb/s
- Idea

  Traffic grooming in networks refers to group low rate traffic into higher speed streams
- Goal
  - Efficient use of the bandwidth
  - Minimize the network cost (in particular equipment)

- **OADM** (Optical Add/Drop Multiplexer)= add and drop wavelength from a fiber
- **ADM** (Add/Drop Multiplexer)= add and drop OC/STM from a wavelength

- **Idea:** Use ADM only at initial and terminal nodes of requests (lightpaths) in order to save as many ADMs as possible

Translation from Telecommunications to Mathematics

Optical Network parameters	Graph Modelization
Optical Network topology	*Digraph G*
Traffic Matrix Instance of requests (i,j)	*Digraph I with arcs (i,j)*
Routing a request r	*Assigning a dipath $P(r)$ in G and a wavelength ω*
Grooming Factor g (a request uses $\frac{1}{g}$ of the available bandwidth on a wavelength ω)	*$\forall$ arc $e \in G$ and $\forall \omega, \exists$ at most g paths of wavelength ω containing arc e*
Requests on wavelength ω	*Subgraph I_ω of I*
ADM on wavelength ω	*Vertex in subgraph I_ω*
Objective : minimize total number of ADMs	*Objective : minimize $\sum_{\omega=1}^{W} \lvert V(I_\omega)\rvert$*

Formalization

Thus, we can state the problem as follows:

The Traffic Grooming Problem for ADMs

Inputs A digraph G (network)
A digraph I (set of requests)
A grooming factor g

Outputs Find for each arc $r \in I$ a path $P(r)$ in G, and a partition of the arcs of I into subgraphs I_ω, $1 \le \omega \le W$, such that $\forall e \in E(G)$ $load(I_\omega, e) \le g$

Objective Minimize $\sum_{\omega=1}^{W} |V(I_\omega)|$, and this minimum is denoted $A(G,I,g)$

The case under study

- In our work we deal with:
  - **Bidirectional Ring topology** $\longrightarrow$ $G = C_N^*$
  - **All-to-all unitary case** $\longrightarrow$ $I = K_N^*$
  - Requests being routed by the **shortest path**
  - We route the opposite requests via the opposite paths
- Given N and g, the optimal solution is noted by **A(g,N)**
- The problem is clearly **NP-complete**
- Example: N=7, g=3 $\longrightarrow$ we use 2 wavelengths and 14 ADMs

Example: $N = 5$ and $g = 2$

Here we decompose K_5 in two ways, both using two wavelengths (colors):

Lower Bounds and some optimal constructions

Theorem (General Lower Bound)

For $G = C_N^*$ and $g = \frac{k(k+1)}{2} + r$, with $0 \le r < k+1$,

$$A(g,N) \ge \frac{N(N-1)}{2}\frac{k+1}{k(k+1)+r}$$

- The general method consists in trying to find explicit constructions that achieve the lower bound
- Example:

Proposition (Optimal construction for $g = 3$)

For $G = C_N^*$, $g = 3$ and $N \equiv 1,5 \pmod{12}$,

$$A(3,N) = \frac{N(N-1)}{4}$$

Example. Let $N = 13$ and $g = 3$

- Let the vertex set of K_{13} be $\{\infty, i, i+6\}$, $i = 0,\ldots,5$
- K_{13} can be decomposed into K_5 and $K_{2,2,2}$:

- We use $\begin{cases} 3\ K_5 : \ \{\infty, i, i+3, i+6, i+9\}, \ \text{with } i = 0,1,2 \\ 4\ K_{2,2,2} : \ \{i,j,k,i+6,j+6,k+6\}, \ \text{with} \\ \qquad (i,j,k) = (0,1,5),(0,2,4),(1,2,3),(3,4,5) \end{cases}$

Lower Bound $\frac{N(N-1)}{4} = 39$ ADMs — This construction $3\cdot 5 + 4\cdot 6 = 39$ ADMs — $\longrightarrow$ **optimal decomposition!**

Case g = 2

Proposition (Tighter Lower Bound for $g = 2$)

For $G = C_N^*$ and $g = 2$,

$$A(2,N) \ge \frac{11N(N-1)}{32}$$

- MILP formulation $\longrightarrow$ it seems this is the right lower bound
- We should use, whenever possible, graphs G_i such that the ratio $\frac{|E_i|}{|V_i|}$ is maximized. Some good graphs for $g = 2$ (ratios $\frac{9}{6}$, $\frac{7}{5}$ and $\frac{7}{5}$):

Case g = 3. Optimal decompositions and tighter lower bounds

- The main result is an **optimal decomposition** for $N \equiv 1,5 \pmod{12}$, using the well know results of the unidirectional case [BCM03]. The graphs used in this decomposition are K_5 and $K_{2,2,2}$.
- We have also found a tighter lower bound for some values of N, taking into account considerations about the degree of the subgraphs:

Proposition (Tighter Lower Bound for g=3 and N=4t+3)

The number of ADMs required in a bidirectional ring with N nodes and grooming factor 3, with $N \equiv 3 \pmod 4$ is lower bounded by the expression

$$A(3, N = 4t+3) \ge \frac{N(3N-1)}{12}$$

Case g = 3. Optimal decompositions and approximations

N	Lower Bound	ADMs of the construction	Ratio $\frac{\text{construction}}{\text{lower bound}}$
1	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4}$	1
3	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 3$	$1 + \frac{\frac{N}{3}-3}{\frac{N(3N-1)}{12}} \xrightarrow{N\to\infty} 1$
5	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4}$	1
7	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 3$	$1 + \frac{\frac{N}{3}-3}{\frac{N(3N-1)}{12}} \xrightarrow{N\to\infty} 1$
9	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4} + 4$	$1 + \frac{4}{\frac{N(N-1)}{4}} \xrightarrow{N\to\infty} 1$
11	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 11$	$1 + \frac{\frac{N}{3}-11}{\frac{N(3N-1)}{12}} \xrightarrow{N\to\infty} 1$

Table: Results of explicit constructions for g=3 and N odd (mod 12)

Case g = 2. Explicit constructions

We have found α-approximations, with α close to 1. For example:

By using this graph beside, we join 2 new nodes (the red ones) to the previous ones (inside the blue circles) obtaining in this way a $\frac{12}{11}$**-approximation**

Summary and Conclusions

- First of all, it is essential to find the right lower bound for each value of g
- By using results of the unidirectional case [BCM03], we have found optimal constructions for these values of g and N:

g	N
3	$N \equiv 1,5 \pmod{12}$
6	$N \equiv 1,7 \pmod{24}$
10	$N \equiv 1,9 \pmod{40}$

- Nevertheless, it is very difficult to find optimal solutions for the general case
- Until now, we have focused mainly in the cases $g = 2$ and $g = 3$, finding in both cases α-approximations, being α either 1 or very close to 1

References

BCM03 J.-C. Bermond, D. Coudert and X. Muñoz, Traffic grooming in Unidirectional WDM Ring Networks: the all-to-all unitary case. ONDM 03, 7th IFIP Working Conference on Optical Network Design and Modelling 2003, pp 1135 – 1153, 3-5 February.

CoMu03 D. Coudert and X. Muñoz, Graph theory and traffic grooming in WDM rings. Recent Res. Devel. Optics 3(2003) 759 – 778

BBC05 J-C. Bermond, L. Braud and D. Coudert. Traffic Grooming on the Path. SIROCCO Vol. 3499 of LNCS, (2005), pp. 34 – 38.

BCY03 J-C. Bermond, D. Coudert and M-L. Yu, On DRC-Covering of K_n by cycles. Journal of Combinatorial Designs Vol. 11 (2003), pp. 110 – 112.

BCC+05 J-C. Bermond, C.J. Colbourn, D. Coudert, G. Ge, A. Ling, and X. Munoz. Traffic grooming in unidirectional WDM rings with grooming ratio $C = 6$. SIAM Journal on Discrete Mathematics, 19 (2005), pp. 523 – 542.

CoWa01 Charles J. Colbourn and Peng-Jun Wan, Minimizing Drop Cost for SONET/WDM Networks with $\frac{1}{8}$ Wavelength Requirements. Networks Vol. 37(2), 107-116 2001.

BeCe03 J-C. Bermond and S. Ceroi. Minimizing SONET ADMs in unidirectional WDM rings with grooming ratio 3. Networks 41 (2003), pp. 83-86.

BeCo03 J.-C. Bermond and D. Coudert. Traffic Grooming in unidirectional WDM ring networks using design theory. IEEE ICC 2003, ON07-3.

Hu02b J. Hu, Traffic Grooming in WDM Ring Networks: A Linear Programming Solution, OSA Journal of Optical Networks 1(11) 2002, pp. 397 – 408.

Appendix C

Paper in 8th ICTON Nottingham

Traffic Grooming in Bidirectional WDM Ring Networks
(extended abstract)

Jean-Claude Bermond[1], David Coudert[2], Xavier Muñoz[3] and Ignasi Sau[4]
[1]Jean-Claude.Bermond@sophia.inria.fr, David.Coudert@sophia.inria.fr, [3]xml@ma4.upc.edu, [4]ignasi@ma4.upc.edu

ABSTRACT

We study the minimization of ADMs (Add-Drop Multiplexers) in Optical WDM Networks with Bidirectional Ring topology considering symmetric shortest path routing and all-to-all unitary requests. We insist on the statement of the problem, which had not been clearly stated before in the bidirectional case. Optimal solutions had not been found up to date. In particular, we study the case $C = 2$ and $C = 3$ (giving either optimal constructions or near-optimal solutions) and the case $C = k(k+1)/2$ (giving optimal decompositions for specific congruence classes of N). We state a general Lower Bound for all the values of C and N, and we improve this Lower Bound for $C=2$ and $C=3$ (when $N=4t+3$). We also include some comments about the simulation of the problem using Linear Programming.

Keywords: SONET over WDM; traffic grooming; ADM; MILP formulation; graph decomposition; combinatorial designs.

1- INTRODUCTION AND STATEMENT OF THE PROBLEM

Traffic grooming in networks refers to grouping low rate traffic into higher speed streams (see [1,2,3]). There are many variants according to the type of network considered (for example, in [4] the Path grooming problem is studied), the constraints used and the parameters to be optimize which give rise to a lot of interesting design problems (graph decomposition).
The objective is to minimize the equipment cost. Among possible criteria, one is to minimize the number of wavelengths used to route all the requests. This leads to the widely studied *loading problem* [5]. Another choice, which is in fact a better approximation of the real equipment cost, is to minimize the number of add/drop locations (called ADMs in SONET terminology) instead of the number of wavelengths. These two problems are proved to be different. Indeed, it is known that even for the simpler network (the unidirectional ring), the number of wavelengths and the number of ADMs cannot be simultaneously minimized [6]. In [7] the bidirectional case is studied, and in [8] a MILP formulation of the traffic grooming in the ring is done.
Let an optical network be represented by a directed graph G (in many cases a symmetric one) on N vertices, for example an unidirectional ring C_N or a bidirectional ring C_N^*. We are given also a *traffic* (or *instance*) *matrix*, that is a family of connection requests represented by a multidigraph I (the number of arcs from i to j corresponding to the number of requests from i to j). It is usual to refer to G as the *physical graph*, whereas I is called *logical graph* (or *request graph*). Satisfying a request r from i to j consists in finding a route (path) $P(r)$ in G and assigning a wavelength to it. The *grooming factor* (or *grooming ratio*) C is the number of unitary requests that can be grouped on a single wavelength, i.e. a request uses only $1/C$ of the bandwith available on a wavelength along its route. In other words, for each arc e of G and for each wavelength w, there are at most C paths using wavelength w and containing arc e.
Let B_w be the subgraph of I that represents the set of instances that use wavelength w. Then the *load* is defined as follows

Definition 1.1 For a subgraph B_w of requests of I, we define the *load* of an edge e of G, $L(B_w,e)$, as the number of requests which are routed through e. i.e.:

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

Work suported by COST action 293: Graphs and Algorithms in Communication Networks. Partially supported by the Spanish Research Council under Projects CICYT, TIC2000-1017 and TIC2001-2171

For each wavelength *w*, an ADM is needed at each node sending or receiving a request on that wavelength. Recall that only one ADM is needed for a node to send or receive various requests on the same wavelength. Therefore The number of ADMs used on wavelength *w* equals the number of vertices in the the subgraph B_w .

Let A_r be the set of paths in *G* through which a request *r* may be routed and let $A = \bigcup_{r \in I} A_r$. The general Traffic Grooming Problem is stated as follows:

Problem 1.1 (The Traffic Grooming Problem)
Input : *A digraph G (network), a digraph I (set of requests), a set A (allowed paths) and a grooming factor C*
Output : *Find for each arc* $r \in I$ *a path* $P(r) \in A_r$*, and a partition of the arcs of I into subgraphs* B_w, $1 \leq w \leq W$*, such that* $\forall e \in E(G)$ $L(B_\omega, e) \leq C$
Objective : *Minimize* $\sum_{w=1}^{W} |V(B_w)|$*, and this minimum is denoted* $A(G, I, A, C)$

In this work we focus on the **Bidirectional Ring Grooming Problem** with **symmetric shortest path routing**, and specifically the **all-to-all unitary case**. i.e., from now on we will consider $G = C_N$ and $I = T_N$, where T_N is a *tournament* containing all the arcs *(i, i+q)*, *i = 0, ... ,N-1, q = 1, ... ,* $\lfloor N/2 \rfloor$ (plus *N/2* arcs of the form *(i, i+N/2)*, if *N* is even).

Remark 1.1 In this case, what we do is minimize the number of ADMs used by the requests following one direction in the cycle, and then double the number of ADMs and the number of wavelengths to compute the total number of ADMs used by the whole set of requests. In this way, we can get rid of the orientation of the requests, because all of them have the same direction. This is the main reason for choosing this routing, besides of its common use in real optical networks. Thus, all the results that we will show take into account only half of the total number of ADMs.

Finally, our problem can be reformulated as follows.

Problem 1.2 (Symmetric Shortest Path routing)
Input : $\vec{C}_N$ *: unidirectional cycle, and a grooming factor C. A set of requests given by a tournament* T_N*. The request (i, j) being routed by the shortest path, and if (i,j) is routed in one direction, (j,i) is routed in the opposite direction*
Output : *Find for each arc* $r \in T_N$ *a path* $P(r)$ *in* $\vec{C}_N$*, and a partition of the arcs of* K_N *into subgraphs* B_w, $1 \leq w \leq W$*, such that* $\forall e \in E(\vec{C}_N)$ $L(B_\omega, e) \leq C$
Objective : *Minimize* $\sum_{w=1}^{W} |V(B_w)|$*, and this minimum is denoted* $A(C, N)$

2- GENERAL LOWER BOUNDS

Let's introduce some notation: consider a valid construction for the Problem and let a_p denote the number of subgraphs of the partition with exactly *p* nodes, *A* the number of ADMs, and *W* the number of subgraphs of the partition.
We have the following equalities:

$$A = \sum_{p=2}^{N} p a_p \tag{1}$$

$$\sum_{p=2}^{N} a_p = W \tag{2}$$

$$\sum_{w=1}^{W} |E_w| = |E| \tag{3}$$

In the particular case where $I = T_N$, we have $|E| = N(N-1)/2$, and we know that

$$W \geq \left\lceil \frac{N^2 - \varepsilon}{8C} \right\rceil, \text{ where } \varepsilon = \begin{cases} 0, & \text{if } N \text{ even} \\ 1, & \text{if } N \text{ odd} \end{cases}$$

To obtain accurate lower bounds we need to bound the value of $|E_\omega|$ for a graph with $|V_\omega| = p$ vertices, satisfying the load constraint.

Definition 2.1 Let $\gamma(C,p)$ be the maximum number of edges of any graph H with p vertices, such that $L(H,e) \le C,\ \forall e \in H$.

Proposition 2.1 (Requests of Shortest Length)
Let $C = k(k+1)/2 + r$, with $0 \le r \le k$, then $\gamma(C,p)$ is given by the expression

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

Using the previous result and equations (1), (2) and (3), we state a general Lower Bound for our Problem:

Theorem 2.1 (General Lower Bound)
For $G = \vec{C}_N$, C = $k(k+1)2 + r$, with $0 \le r \le k$, and if all the requests cannot be put in a single subgraph:

$$A(C,N) \ge \left\lceil \frac{N(N-1)}{2} \frac{k+1}{k(k+1)+r} \right\rceil$$

It is easy to se that for *C=1* the lower bound can always be reached using the decomposition in cycles that can be found in [10].

3- CASE C=2

We have improved the previous Lower Bound when the grooming factor is equal to 2.

Proposition 3.1 (Tighter Lower Bound for C=2)
For $G = \vec{C}_N$ and C = 2,

$$A(2,N) \ge \left\lceil \frac{11N(N-1)}{32} \right\rceil$$

To get an idea about the validity of this Lower Bound, we have formulated the Problem using MILP, taking into account equations (1), (2) and (3). A graph of the ratio *ADM_simplex/ADM_LB* vs *Log(N)* is drawn in Fig. 1, for *N>14*. We can observe in this graph that the ratio tends to 1 when *N* increases. Notice that a solution given by MILP does not imply the existence of a valid decomposition, since the load of edges is not taken into account.

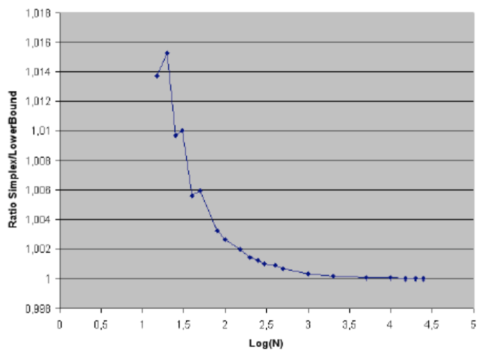


Figure 1. Simplex/LB vs Log(N)

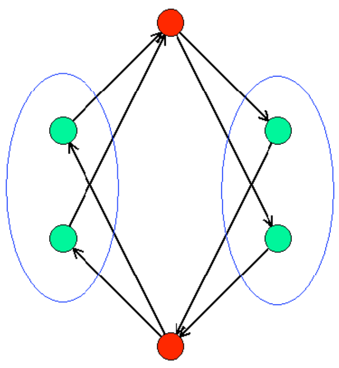


Figure 2. Graph used in the 12/11-approx

We have found explicit decompositions which are near-optimal solutions for *C=2*. In particular, we proved that for odd values of *N*, T_N can be decomposed into triangles and subgraphs isomorphic to the one depicted in Fig. 2 in such a way that the load of every edge is less or equal to 2 and the number of ADMs given by such decomposition is less than 12/11 times the lower bound.

4- CASE C=3

Similarly to the previous case, we have improved the Lower Bound when *N=4t+3*, and we have found either optimal solutions (see Section 5) or 1-approxs (extending the Steiner triples construction in [10]) for all the odd values of *N*, as we can see in Table 1. The extension to the even values of *N* should be straightforward.

$N \pmod{12}$	Lower Bound	ADMs of the explicit construction
1, 5	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4}$
3, 7	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 3$
9	$\frac{N(N-1)}{4}$	$\frac{N(N-1)}{4} + 4$
11	$\frac{N(3N-1)}{12}$	$\frac{N(3N-1)}{12} + \frac{N}{3} - 11$

Table 1. Explicit constructions for C=3 and N odd

5- CASE C=k(k+1)/2

Using the results of the unidirectional case, we have found optimal constructions for several values of *C* and infinite families of values of *N*. The main idea is to take known decompositions of the unidirectional case (using combinatorial designs that are in fact BIBDs, see [1] and [11]), and then split each vertex (except one) to obtain graphs that become optimal for the bidirectional case (under criterion of Proposition 2.1). In Table 2 we sum up the values for which we have obtained optimal solutions.

C	N
3	$N \equiv 1, 5 \pmod{12}$
6	$N \equiv 1, 7 \pmod{24}$
10	$N \equiv 1, 9 \pmod{40}$

Table 2. Optimal solutions

6- CONCLUSIONS

In this paper we have formally stated the Traffic Grooming Problem considering symmetric shortest path routing and all-to-all unitary requests, and we have given a general Lower Bound. It would be interesting to try to improve this LB for other values of *C* (besides 2 and 3), but it seems to be a difficult problem.
Finding other optimal constructions will lead to exciting graph decomposition problems, where tools from combinatorial design are strongly needed. Another approach to the problem consists in finding good approximations, which sometimes (but not always) has more practical value.
Other routings and topologies could be considered, and we are convinced that many of the results that we sum up in this paper can be extended to similar problems.

REFERENCES

[1] J.-C. Bermond, D. Coudert and X. Muñoz: Traffic grooming in Unidirectional WDM Ring Networks: the all-to-all unitary case. *ONDM 03, 7th IFIP Working Conference on Optical Network Design and Modelling 2003, pp 1135 - 1153, 3-5 February.*

[2] J.-C. Bermond, *et al*: Traffic grooming in unidirectional WDM rings with grooming ratio C = 6. *SIAM Journal on Discrete Mathematics, 19 (2005), pp. 523 - 542.*

[3] J.-C. Bermond and D. Coudert: Traffic Grooming in unidirectional WDM ring networks using design theory. *IEEE ICC 2003, ON07-3.*

[4] J.-C. Bermond, L. Braud and D. Coudert: Traffic Grooming on the Path. *SIROCCO Vol. 3499 of LNCS, (2005), pp. 34 - 38.*

[5] B. Beauquier, *et al*: Graph problems arising from wavelength-routing in all-optical networks. *In IEEE Workshop on Optics and Computer Science, Geneva, 1997*

[6] A. Chiu and E.H. Modiano: Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. *IEEE/OSA Journal of Lightwave Technology, 2000, 18(1) :2-12*

[7] Charles J. Colbourn and Peng-Jun Wan: Minimizing Drop Cost for SONET/WDM Networks with 1/8 Wavelength Requirements. *Networks Vol. 37(2), 107-116 2001.*

[8] J. Hu: Traffic Grooming in WDM Ring Networks: A Linear Programming Solution, *OSA Journal of Optical Networks 1(11) 2002, pp. 397 - 408*

[9] J.-C. Bermond, L. Chacon, D. Coudert, and F. Tillerot: Cycle Covering. *International Colloquium on Structural Information and Communication Complexity -- SIROCCO, Proceedings in Informatics 11, Vall de Nuria, Spain, pages 21-34, 27-29, June 2001*

[10] Charles J. Colbourn and Jeffrey H. Dinitz: The CRC Handbook of Combinatorial Designs. *1996 by CRC Press, Inc*

References

[1] www.mathworld.com.

[2] www.wikipedia.com.

[3] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Pérennes, and U. Vaccaro. Graph problems arising from wavelength-routing in all-optical networks. In *IEEE Workshop on Optics and Computer Science*, Geneva, Switzerland, 1997.

[4] J.-C. Bermond, L. Braud, and D. Coudert. Traffic Grooming on the Path. In *12th International Colloquium on Structural Information and Communication Complexity – SIROCCO*, number 3499 in 24-26, pages 34–48. LNCS 3499, May 24-26 2005.

[5] J.-C. Bermond and S. Ceroi. Minimizing SONET ADMs in unidirectional WDM rings with grooming ratio 3. *Networks*, 41(2) :83–86, 2003.

[6] J.-C. Bermond, L. Chacon, D. Coudert, and F. Tillerot. Cycle Covering. In *International Colloquium on Structural Information and Communication Complexity – SIROCCO*, volume 11, pages 21–34, 27–29, Vall de Núria, Spain, June 2001. Carleton Scientific.

[7] J.-C. Bermond, C. Colbourn, A. Ling, and M.-L. Yu. Grooming in unidirectional rings : $K_4 - e$ designs. *Discrete Mathematics, Lindner's Volume*, 284(1-3) :57–62, 2004.

[8] J.-C. Bermond, C. J. Colbourn, D. Coudert, G. Ge, A. C. 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.

[9] J.-C. Bermond, M. Cosnard, D. Coudert, and S. Pérennes. Optimal Solution of the Maximum All Request Path Grooming Problem. In *Advanced International Conference on Telecommunications (AICT)*. IEEE, 2006.

[10] J.-C. Bermond, D. Coudert, and X. Muñoz. Traffic Grooming in Unidirectional WDM Ring Networks : the all-to-all unitary case. In *ONDM 03, 7th IFIP Working Conference on Optical Network Design and Modelling*, pages 1135–1153, February 3-5 2003.

[11] J.-C. Bermond, D. Coudert, and M.-L. Yu. On DRC-Covering of K_n by Cycles. *Journal of Combinatorial Designs*, 11 :100–112, 2003.

[12] 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*, pages 1961–1971, 2000.

[13] A. Chiu and E. Modiano. Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. *IEEE/OSA Journal of Lightwave Technology*, 18(1) :2–12, 2000.

[14] T. Y. Chow and P. J. Lin. The Ring Grooming Problem. *Networks*, pages 194–202, January 2004.

[15] C. J. Colbourn and J. H.Dinitz. *The CRC Handbook of Combinatorial Designs.* CRC Press, Inc, 1996.

[16] C. J. Colbourn and A. C. Ling. Graph decompositions with application to wavelength add-drop multiplexing for minimizing SONET ADMs. *Discrete Mathematics 261*, pages 141–156, 2003.

[17] C. J. Colbourn, G. Quattrocchi, and V. R. Syrotiuk. Grooming for Two-Period Optical Networks. To appear.

[18] C. J. Colbourn and P.-J. Wan. Minimizing Drop Cost for SONET/WDM Networks with $\frac{1}{8}$ Wavelength Requirements. *Networks*, 37(2) :107–116, 2001.

[19] F. Comellas, J. Fàbrega, and O. S. A. Sànchez. *Matemática Discreta.* Edicions UPC, 2001.

[20] T. Cormen, C. Leiserson, and R. Rivest. *Introduction à l'algorithmique.* DUNOD, Paris, 1994.

[21] S. Cosares and I. Saniee. An optimization problem related to balancing loads on SONET rings. *Telecom Systems 3*, pages 165–181, 1994.

[22] D. Coudert and X. Muñoz. Graph theory and traffic grooming in WDM rings. *Recent Res. Devel. Optics*, 3 :759–778, 2003.

[23] R. Diestel. *Graph Theory.* Springer-Verlag, 2005.

[24] R. Dutta and N. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks*, 1(1) :73–89, 2000.

[25] R. Dutta and N. Rouskas. Traffic grooming in WDM networks : Past and future. Technical report, CSC TR-2002-08, NCSU, 2002.

[26] P. Erdös and J. Spencer. *Probabilistic methods in combinatorics.* Academic Press, New York, 1974.

[27] M. Flammini, L. Moscardelli, M. Shalom, and S. Zaks. Approximating the Traffic Grooming Problem. In *Algorithms and Computation : 16th International Symposium, ISAAC 2005*, pages 915–924, Sanya, Hainan, China, December 19-21 2005.

[28] A. Frank, T. Nishizeki, N. Saito, H. Suzuki, and E. Tardos. Algorithms for routing around a rectangle. *Discrete Appl Math 40*, pages 363–378, 1992.

[29] O. Gerstel, P. Lin, and G. Sasaki. Wavelength assignment in a WDM ring to minimize cost of embedded SONET rings. *IEEE Infocom*, pages 94–101, 1998.

[30] O. Gerstel, P. Lin, and G. Sasaki. Combined WDM and SONET network design. In *IEEE Infocom*, pages 734–743, 1999.

[31] P. Harshavardhana, P. Johri, and R. Nagarajan. A note on weight-based load balancing on SONET rings. *Telecom Systems 6*, pages 237–239, 1996.

[32] J. H.Dinitz and D. R.Stinson. *Contemporary Design Theory. A Collection of Surveys.* Wiley-Interscience Series in Discrete Mathematics and Optimization, 1992.

[33] J. Hu. Optimal traffic grooming for wavelength-division-multiplexing rings with all-to-all uniform traffic. *OSA Journal of Optical Networks*, 1(1) :32–42, 2002.

[34] J. Hu. Traffic Grooming in WDM Ring Networks : A Linear Programming Solution. *OSA Journal of Optical Networks*, 1(11) :397–408, 2002.

[35] S. Huang, R. Dutta, and G. N. Rouskas. Traffic Grooming in Path, Star, and Tree Networks : Complexity, Bounds, and Algorithms. In *OPTICOMM*, 2003.

[36] S. Khanna. A polynomial time approximation scheme for the SONET ring loading problem. *Bell Labs Tech J 2*, pages 36–41, 1997.

[37] H. Kopka and P. W. Daly. *A Guide to LaTeX 2ε.* Addison-Wesley, 2nd edition, 1995.

[38] L. Lamport. *LaTeX. User's Guide and Reference Manual.* Addison-Wesley, 2nd edition, 1994.

[39] C. Lindner and C. Rodger. *Design Theory.* CRC Press LLC, 1997.

[40] E. Modiano and P. Lin. Traffic grooming in WDM networks. *IEEE Communications Magazine*, 39(7) :124–129, 2001.

[41] Y. Myung, H. Kim, and D. Tcha. Optimal load balancing on SONET bidirectional rings. *Operat Res 45*, pages 148–152, 1997.

[42] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[43] A. Schrijver, P. Seymour, and P. Winkler. The ring loading problem. *SIAM J Discrete Math 11*, pages 1–14, 1998.

[44] A. Somani. Survivable traffic grooming in WDM networks. In I. D. Gautam, editor, *Broad band optical fiber communications technology - BBOFCT*, pages 17–45, Jalgaon, India. Nirtali Prakashan, 2001. Invited paper.

[45] A. Street and B. Street. *Combinatorics of Experimental Design.* Oxford University Press, 1987.