Optimization in Graphs Under Degree Constraints.

Application to Telecommunication Networks

Ignasi Sau Valls
Mascotte – MA4

Advisors:
Jean-Claude Bermond, David Coudert, Xavier Muñoz

October 16, 2009

Outline of the talk

Traffic grooming

Degree-constrained subgraph problems

Outline of the talk

Traffic grooming		Degree-constrained subgraph problems

- Motivation
- Overview of the results

- Motivation
- Overview of the results

Outline of the talk

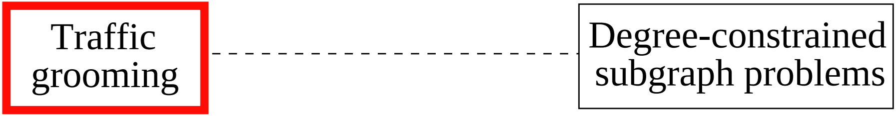
Traffic grooming

- Motivation
- Overview of the results
- Some details on one aspect

Degree-constrained subgraph problems

- Motivation
- Overview of the results
- Some details on one aspect

Outline of the talk



General idea

- WDM (Wavelength Division Multiplexing) networks
  - 1 wavelength (or frequency) = up to 40 Gb/s
  - 1 fiber = hundreds of wavelengths = Tb/s

- Traffic grooming consists in packing low-speed traffic flows into higher speed streams

  ⟶ we allocate the same wavelength to several low-speed requests (TDM, Time Division Multiplexing)

- Objectives:
  - Better use of bandwidth
  - Reduce the equipment cost (mostly given by electronics)

General idea

- WDM (Wavelength Division Multiplexing) networks
  - 1 wavelength (or frequency) = up to 40 Gb/s
  - 1 fiber = hundreds of wavelengths = Tb/s

- Traffic grooming consists in packing low-speed traffic flows into higher speed streams

  $\longrightarrow$ we allocate the same wavelength to several low-speed requests (TDM, Time Division Multiplexing)

- Objectives:
  - Better use of bandwidth
  - Reduce the equipment cost (mostly given by electronics)

General idea

- WDM (Wavelength Division Multiplexing) networks
  - 1 wavelength (or frequency) = up to 40 Gb/s
  - 1 fiber = hundreds of wavelengths = Tb/s
- Traffic grooming consists in packing low-speed traffic flows into higher speed streams

  $\longrightarrow$ we allocate the same wavelength to several low-speed requests (TDM, Time Division Multiplexing)
- Objectives:
  - Better use of bandwidth
  - Reduce the equipment cost (mostly given by electronics)

Definitions

- **Request** (i, j): two vertices (i, j) that want to exchange (low-speed) traffic

- **Grooming factor** C:

$$C = \frac{\text{Capacity of a wavelength}}{\text{Capacity used by a request}}$$

⋆ Typical values of the grooming factor:
SDH: $4, 16, 64, 256, \ldots$
SONET: $3, 12, 48, \ldots$

Example:
Capacity of one wavelength $= 2.5\ Gb/s$
Capacity used by a request $= 640\ Mb/s$ $\Rightarrow C = 4$

- **load** of an arc in a wavelength: number of requests using this arc in this wavelength ($\leq C$)

Definitions

- **Request** (i, j): two vertices (i, j) that want to exchange (low-speed) traffic
- **Grooming factor** C**:**

$$C = \frac{\text{Capacity of a wavelength}}{\text{Capacity used by a request}}$$

⋆ Typical values of the grooming factor:

SDH: $4, 16, 64, 256, \ldots$

SONET: $3, 12, 48, \ldots$

Example:
Capacity of one wavelength $= 2.5\ Gb/s$
Capacity used by a request $= 640\ Mb/s$ $\Rightarrow\ C = 4$

- **load** of an arc in a wavelength: number of requests using this arc in this wavelength ($\leq C$)

Definitions

- **Request** (i,j): two vertices (i,j) that want to exchange (low-speed) traffic
- **Grooming factor** C**:**

$$C = \frac{\text{Capacity of a wavelength}}{\text{Capacity used by a request}}$$

⋆ Typical values of the grooming factor:
SDH: $4, 16, 64, 256, \ldots$
SONET: $3, 12, 48, \ldots$

Example:
Capacity of one wavelength $= 2.5\ Gb/s$
Capacity used by a request $= 640\ Mb/s$ $\Rightarrow\ C = 4$

- **load** of an arc in a wavelength: number of requests using this arc in this wavelength ($\leq C$)

Definitions

- **Request** (i, j): two vertices (i, j) that want to exchange (low-speed) traffic
- **Grooming factor** C**:**

$$C = \frac{\text{Capacity of a wavelength}}{\text{Capacity used by a request}}$$

⋆ Typical values of the grooming factor:

SDH: $4, 16, 64, 256, \ldots$

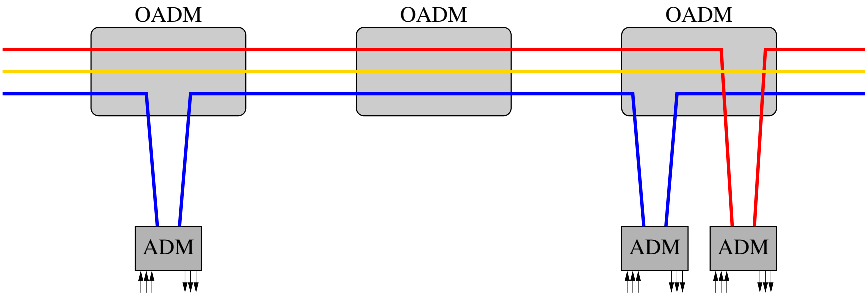
SONET: $3, 12, 48, \ldots$

Example:
Capacity of one wavelength $= 2.5\ Gb/s$
Capacity used by a request $= 640\ Mb/s$ $\Rightarrow C = 4$

- **load** of an arc in a wavelength: number of requests using this arc in this wavelength ($\leq C$)

ADM and OADM

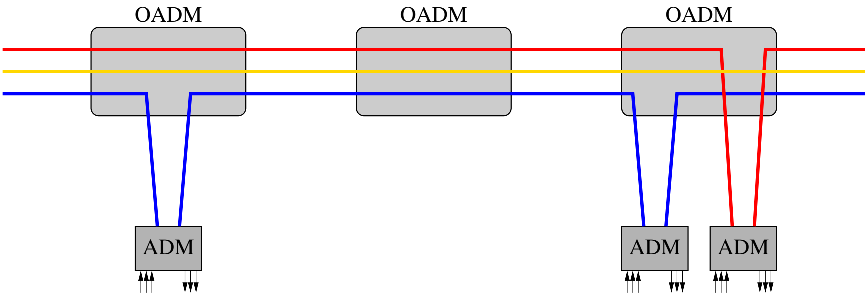
- **OADM** (Optical Add/Drop Multiplexer)= insert/extract a wavelength to/from an optical fiber
- **ADM** (Add/Drop Multiplexer)= insert/extract an OC/STM (electric low-speed signal) to/from a wavelength



- We want to **minimize the number of ADMs**
- We need to use an **ADM only at the endpoints of a request** (lightpaths) in order to save as many ADMs as possible

ADM and OADM

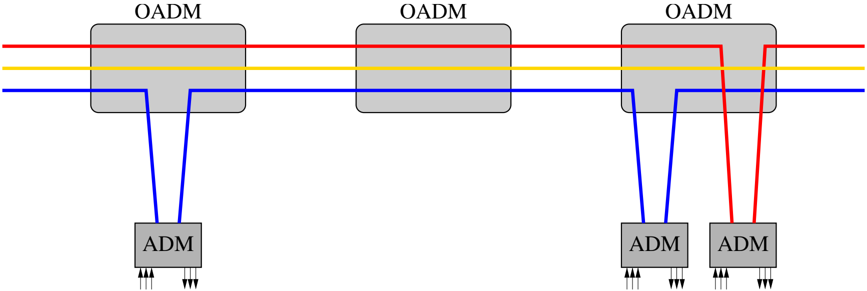
- **OADM** (Optical Add/Drop Multiplexer)= insert/extract a wavelength to/from an optical fiber
- **ADM** (Add/Drop Multiplexer)= insert/extract an OC/STM (electric low-speed signal) to/from a wavelength



- We want to **minimize the number of ADMs**
- We need to use an **ADM only at the endpoints of a request** (lightpaths) in order to save as many ADMs as possible

ADM and OADM

- **OADM** (Optical Add/Drop Multiplexer)= insert/extract a wavelength to/from an optical fiber
- **ADM** (Add/Drop Multiplexer)= insert/extract an OC/STM (electric low-speed signal) to/from a wavelength



- We want to **minimize the number of ADMs**
- We need to use an **ADM only at the endpoints of a request** (lightpaths) in order to save as many ADMs as possible

To fix ideas...

- Model:

Topology	$\longrightarrow$	graph G
Request set	$\longrightarrow$	graph R
Grooming factor	$\longrightarrow$	integer C
Wavelength	$\longrightarrow$	**Subgraph** of R
Requests in a **wavelength**	$\longrightarrow$	edges in a **subgraph** of R
ADM in a wavelength	$\longrightarrow$	vertex in a subgraph of R

- A fundamental case is when $G = \overrightarrow{C}_n$ (**unidirectional ring**)
- It is also natural to consider **symmetric requests**

To fix ideas...

- Model:

Topology	$\rightarrow$	graph G
Request set	$\rightarrow$	graph R
Grooming factor	$\rightarrow$	integer C
Wavelength	$\rightarrow$	**Subgraph** of R
Requests in a **wavelength**	$\rightarrow$	edges in a **subgraph** of R
ADM in a wavelength	$\rightarrow$	vertex in a subgraph of R

- A fundamental case is when $G = \overrightarrow{C}_n$ (**unidirectional ring**)
- It is also natural to consider **symmetric requests**

To fix ideas...

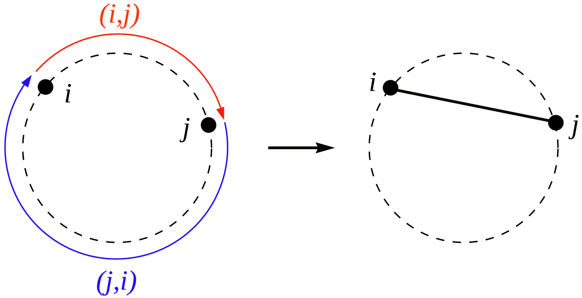
- Model:

Topology	$\longrightarrow$	graph G
Request set	$\longrightarrow$	graph R
Grooming factor	$\longrightarrow$	integer C
Wavelength	$\longrightarrow$	**Subgraph** of R
Requests in a **wavelength**	$\longrightarrow$	edges in a **subgraph** of R
ADM in a wavelength	$\longrightarrow$	vertex in a subgraph of R

- A fundamental case is when $G = \overrightarrow{C}_n$ (**unidirectional ring**)
- It is also natural to consider **symmetric requests**

Unidirectional ring with symmetric requests

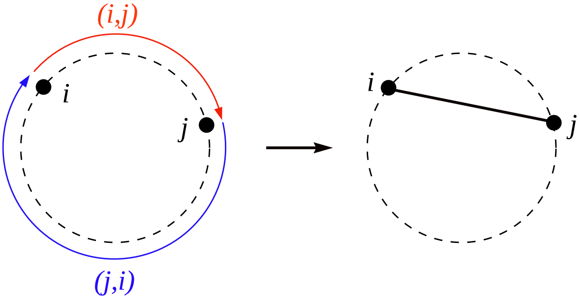
- **Symmetric requests**: whenever there is the request (i, j), there is also the request (j, i).



- W.l.o.g. requests (i, j) and (j, i) are in the same subgraph
$\rightarrow$ each pair of symmetric requests induces load 1
$\rightarrow$ **grooming factor C** $\Leftrightarrow$ **each subgraph has $\leq C$ edges.**

Unidirectional ring with symmetric requests

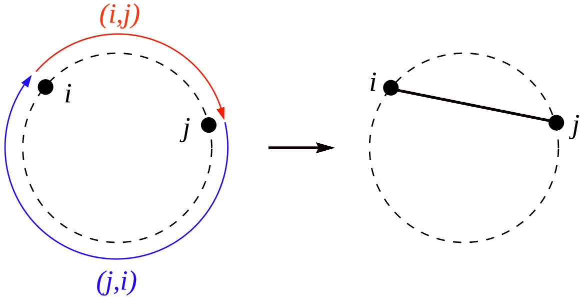
- **Symmetric requests**: whenever there is the request (i, j), there is also the request (j, i).



- W.l.o.g. requests (i, j) and (j, i) are in the same subgraph
  $\rightarrow$ each pair of symmetric requests induces load 1
  $\rightarrow$ **grooming factor C $\Leftrightarrow$ each subgraph has $\leq C$ edges.**

Unidirectional ring with symmetric requests

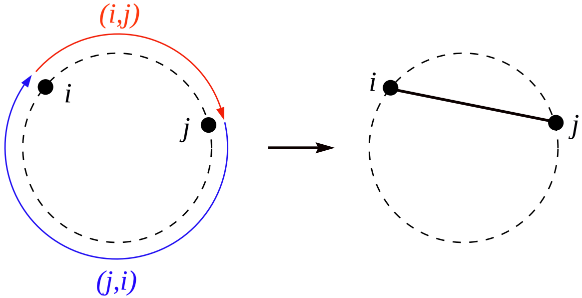
- **Symmetric requests**: whenever there is the request (i, j), there is also the request (j, i).



- W.l.o.g. requests (i, j) and (j, i) are in the same subgraph
  $\rightarrow$ each pair of symmetric requests induces load 1
  $\rightarrow$ **grooming factor C $\Leftrightarrow$ each subgraph has $\leq C$ edges.**

Unidirectional ring with symmetric requests

- **Symmetric requests**: whenever there is the request (i, j), there is also the request (j, i).



- W.l.o.g. requests (i, j) and (j, i) are in the same subgraph
  $\rightarrow$ each pair of symmetric requests induces load 1
  $\rightarrow$ **grooming factor C** $\Leftrightarrow$ **each subgraph has $\leq C$ edges.**

Statement of the problem

Traffic Grooming in Unidirectional Rings (with symmetric requests)

Input An *undirected* graph R on n nodes (request set);
A grooming factor C.

Output A partition of $E(R)$ into subgraphs
$R_1, \ldots, R_W$ with $|E(R_i)| \leq C$, i=1,...,W.

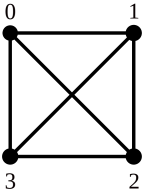
Objective Minimize $\sum_{i=1}^{W} |V(R_i)|$.

Example (unidirectional ring with symmetric requests)

$n = 4$

$R = K_4$

$C = 3$

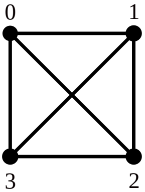


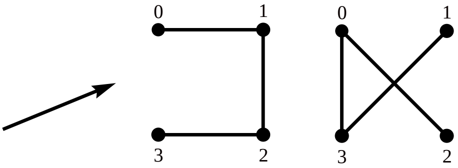
Example (unidirectional ring with symmetric requests)

$n = 4$

$R = K_4$

$C = 3$



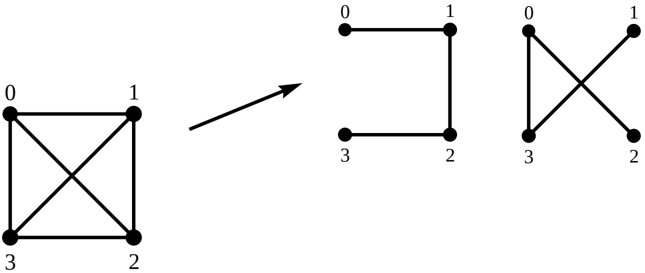


Example (unidirectional ring with symmetric requests)

$n = 4$

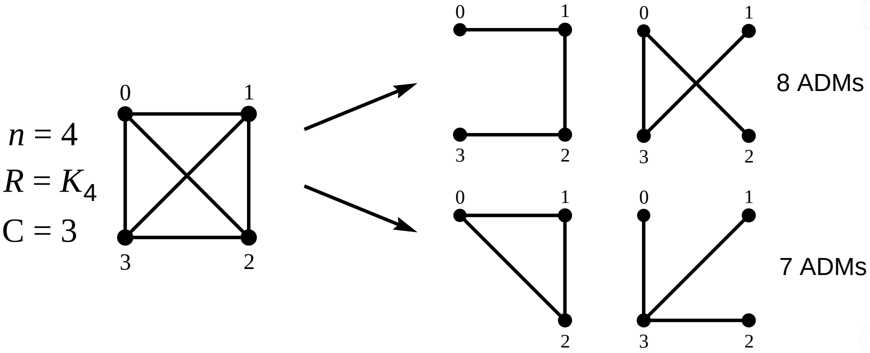
$R = K_4$

$C = 3$

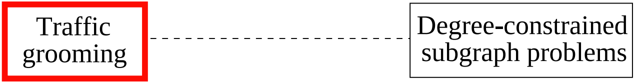


8 ADMs

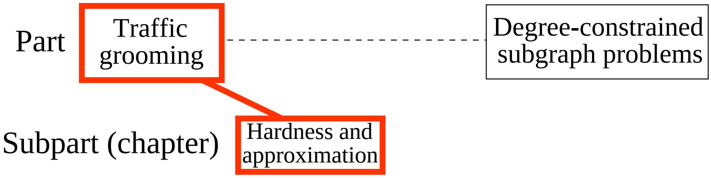
Example (unidirectional ring with symmetric requests)



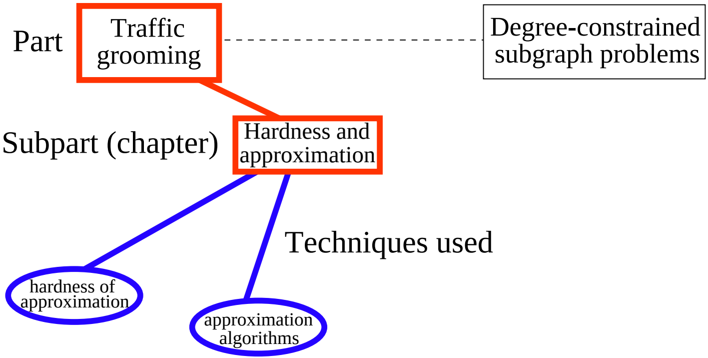
Graph of the thesis



Graph of the thesis



Graph of the thesis



Preliminaries: approximation algorithms

- Given a (typically NP-hard) **minimization** problem Π, ALG is an **α-approximation algorithm** for Π (with $\alpha \geq 1$) if for any instance *I* of Π,

$$ALG(I) \leq \alpha \cdot OPT(I).$$

- **Class APX (Approximable):**

  an NP-hard optimization problem is in APX if it can be approximated within a constant factor.

  Example: MINIMUM VERTEX COVER has a 2-approximation.

- **Class PTAS (Polynomial-Time Approximation Scheme):**

  an NP-hard optimization problem is in PTAS if it can be approximated within a constant factor $1 + \varepsilon$, for all $\varepsilon > 0$
  (the best one can hope for an NP-hard problem).

  Example: MAXIMUM KNAPSACK.

Preliminaries: approximation algorithms

- Given a (typically NP-hard) **minimization** problem Π, ALG is an **α-approximation algorithm** for Π (with $\alpha \geq 1$) if for any instance I of Π,

$$ALG(I) \leq \alpha \cdot OPT(I).$$

- **Class APX (Approximable):**

  an NP-hard optimization problem is in APX if it can be approximated within a constant factor.

  Example: MINIMUM VERTEX COVER has a 2-approximation.

- **Class PTAS (Polynomial-Time Approximation Scheme):**

  an NP-hard optimization problem is in PTAS if it can be approximated within a constant factor $1 + \varepsilon$, for all $\varepsilon > 0$
  (the best one can hope for an NP-hard problem).

  Example: MAXIMUM KNAPSACK.

Preliminaries: approximation algorithms

- Given a (typically NP-hard) **minimization** problem Π, ALG is an α**-approximation algorithm** for Π (with $\alpha \geq 1$) if for any instance I of Π,

$$ALG(I) \leq \alpha \cdot OPT(I).$$

- **Class APX (Approximable):**

  an NP-hard optimization problem is in APX if it can be approximated within a constant factor.

  Example: MINIMUM VERTEX COVER has a 2-approximation.

- **Class PTAS (Polynomial-Time Approximation Scheme):**

  an NP-hard optimization problem is in PTAS if it can be approximated within a constant factor $1 + \varepsilon$, for all $\varepsilon > 0$
  (the best one can hope for an NP-hard problem).

  Example: MAXIMUM KNAPSACK.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

★ **Open problem:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

★ **Open problem:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

★ **Open problem:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

★ **Open problem:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

★ **Open problem:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
[Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
[Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
(the proof assumes a bounded number of wavelengths)
[Shalom, Unger, and Zaks. *FUN'07*]

★ **Open problem:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

✓ ~~**Open problem**~~**:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Hardness of RING TRAFFIC GROOMING

1. **NP-complete** if C is part of the input
   [Chiu and Modiano. *IEEE JLT'00*]
2. **Not in APX** if C is part of the input
   [Huang, Dutta, and Rouskas. *IEEE JSAC'06*]
3. Remains **NP-complete** for fixed $C \geq 1$
   (the proof assumes a bounded number of wavelengths)
   [Shalom, Unger, and Zaks. *FUN'07*]

✓ ~~**Open problem**~~**:** inapproximability for fixed C?

Conjecture: Not in PTAS for fixed C.

[Wan, Calinescu, Liu, and Frieder. *IEEE JSAC'00*]
[Chow and Lin. *Networks'04*]

Theorem (Amini, Pérennes, and S.)

RING TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 1$.
PATH TRAFFIC GROOMING *is not in* PTAS *for any fixed* $C \geq 2$.

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

★ **Open problem:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

★ **Open problem:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

★ **Open problem:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

★ **Open problem:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

★ **Open problem:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

✓ ~~**Open problem**~~**:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of Ring Traffic Grooming

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

✓ ~~**Open problem**~~**:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

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

Outline of the algorithm:

1. partition the requests into groups of similar length
2. in each group, extract "dense" subgraphs greedily using an algorithm for the Dense k-subgraph problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

✓ ~~**Open problem**~~**:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$*.*

Outline of the algorithm:

1. partition the requests into groups of **similar length** [**factor** log **n**]
2. in each group, extract "dense" subgraphs greedily using an algorithm for the DENSE k-SUBGRAPH problem

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

✓ ~~**Open problem**~~**:** approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$.

Outline of the algorithm:

1. partition the requests into groups of **similar length** [**factor** log **n**]
2. in each group, extract "dense" subgraphs **greedily** using an algorithm for the DENSE k-SUBGRAPH problem **[factor** log n**]**

Approximation of RING TRAFFIC GROOMING

1. $\sqrt{C}$-approximation is trivial (in poly-time in both n and C)
2. $\mathcal{O}(\log C)$-approximation algorithm, with running time $\mathcal{O}(n^C)$ [Flammini et al. *ISAAC'05, JDA'08*]
3. But in backbone networks, it is usually the case that $C \geq n$.

✓ ~~Open problem~~: approximation algorithm in poly-time in **both** C and n, and with approximation factor **independent of** C.

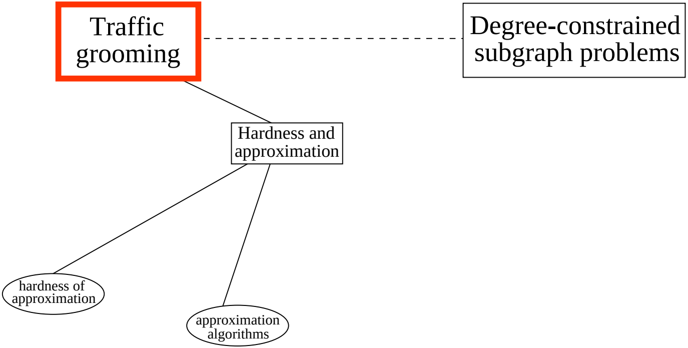
Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$*.*

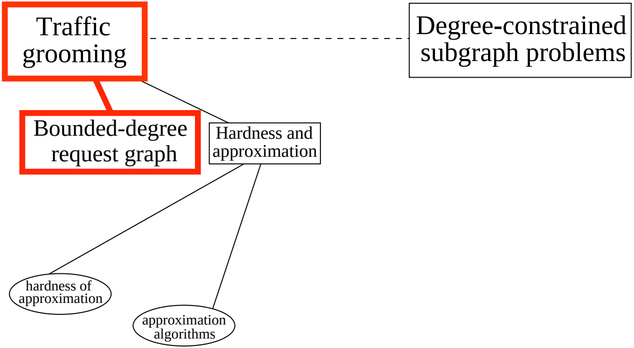
Outline of the algorithm:

1. partition the requests into groups of **similar length** [**factor** log **n**]
2. in each group, extract "dense" subgraphs **greedily** using an algorithm for the DENSE k-SUBGRAPH problem **[factor** $\log n$**]** **[factor** $n^{1/3}$**]**

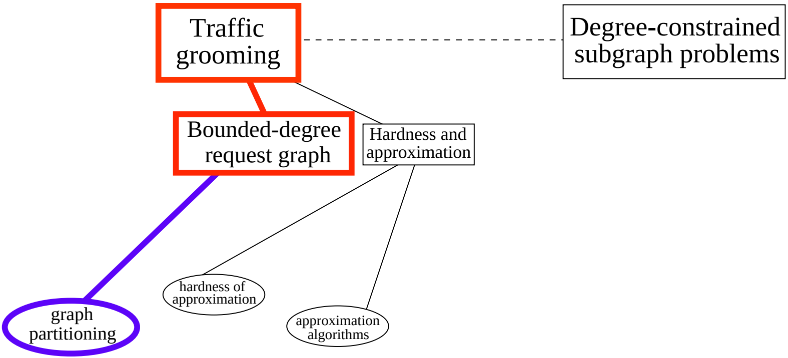
Graph of the thesis



Graph of the thesis



Graph of the thesis



New model of traffic grooming

- In the literature so far:
  place ADMs at nodes for a **fixed** **request graph.**
  → placement of ADMs **a posteriori.**

- **New model** [With Xavier Muñoz]:
  place the ADMs at nodes such that the network can support **any request graph with maximum degree at most** Δ**.**
  → placement of ADMs **a priori.**

- As the network must support any degree-bounded graph, due to symmetry we place the **same number of ADMs at each node.**

- The objective is then to minimize this number.

New model of traffic grooming

- In the literature so far:
  place ADMs at nodes for a **fixed request graph.**
  → placement of ADMs **a posteriori**.

- **New model** [With Xavier Muñoz]:
  place the ADMs at nodes such that the network can support **any request graph with maximum degree at most** Δ**.**
  → placement of ADMs **a priori**.

- As the network must support any degree-bounded graph, due to symmetry we place the **same number of ADMs at each node.**

- The objective is then to minimize this number.

New model of traffic grooming

- In the literature so far:
  place ADMs at nodes for a **fixed request graph.**
  → placement of ADMs **a posteriori**.

- **New model** [With Xavier Muñoz]:
  place the ADMs at nodes such that the network can support **any request graph with maximum degree at most Δ.**
  → placement of ADMs **a priori.**

- As the network must support any degree-bounded graph, due to symmetry we place the **same number of ADMs at each node.**

- The objective is then to minimize this number.

New model of traffic grooming

- In the literature so far:
  place ADMs at nodes for a **fixed request graph.**
  → placement of ADMs **a posteriori**.

- **New model** [With Xavier Muñoz]:
  place the ADMs at nodes such that the network can support **any request graph with maximum degree at most Δ.**
  → placement of ADMs **a priori**.

- As the network must support any degree-bounded graph, due to symmetry we place the **same number of ADMs at each node.**

- The objective is then to minimize this number.

New model of traffic grooming

- In the literature so far:
  place ADMs at nodes for a **fixed request graph.**
  → placement of ADMs **a posteriori**.
- **New model** [With Xavier Muñoz]:
  place the ADMs at nodes such that the network can support **any request graph with maximum degree at most Δ.**
  → placement of ADMs **a priori**.
- As the network must support any degree-bounded graph, due to symmetry we place the **same number of ADMs at each node.**
- The objective is then to minimize this number.

New model of traffic grooming

- In the literature so far:
  place ADMs at nodes for a **fixed request graph.**
  → placement of ADMs **a posteriori**.
- **New model** [With Xavier Muñoz]:
  place the ADMs at nodes such that the network can support **any request graph with maximum degree at most Δ.**
  → placement of ADMs **a priori**.
- As the network must support any degree-bounded graph, due to symmetry we place the **same number of ADMs at each node.**
- The objective is then to minimize this number.

The parameter $M(C, \Delta)$

- Δ-graph: graph with maximum degree at most Δ.
- C-edge partition of G: partition of $E(G)$ into subgraphs with $\leq C$ edges.
- The problem is equivalent to determining the following parameter:
- Therefore, we focus on determining $M(C, \Delta)$.
- W.l.o.g. we can assume that R has **regular degree** Δ.

Proposition (Lower Bound – Muñoz and S.)

For all $C, \Delta \geq 1$, $M(C, \Delta) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$.

The parameter $M(C, \Delta)$

- Δ-graph: graph with maximum degree at most Δ.
- C-edge partition of G: partition of $E(G)$ into subgraphs with $\leq C$ edges.
- The problem is equivalent to determining the following parameter:

- Therefore, we focus on determining $M(C, \Delta)$.
- W.l.o.g. we can assume that R has **regular degree** Δ.

Proposition (Lower Bound – Muñoz and S.)

For all $C, \Delta \geq 1$, $M(C, \Delta) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$.

The parameter $M(C, \Delta)$

- Δ-graph: graph with maximum degree at most Δ.
- C-edge partition of G: partition of $E(G)$ into subgraphs with $\leq C$ edges.
- The problem is equivalent to determining the following parameter:

$M(C, \Delta)$: smallest integer M s.t. **any** Δ-graph has a C-edge-partition s.t. each vertex appears in $\leq M$ subgraphs.

- Therefore, we focus on determining $M(C, \Delta)$.
- W.l.o.g. we can assume that R has **regular degree** Δ.

Proposition (Lower Bound – Muñoz and S.)

For all $C, \Delta \geq 1$, $M(C, \Delta) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$.

The parameter $M(C, \Delta)$

- Δ-graph: graph with maximum degree at most Δ.
- C-edge partition of G: partition of $E(G)$ into subgraphs with $\leq C$ edges.
- The problem is equivalent to determining the following parameter:

> $M(C, \Delta)$: smallest integer M s.t. **any** Δ-graph has a C-edge-partition s.t. each vertex appears in $\leq M$ subgraphs.

- Therefore, we focus on determining $M(C, \Delta)$.
- W.l.o.g. we can assume that R has **regular degree** Δ.

Proposition (Lower Bound – Muñoz and S.)

For all $C, \Delta \geq 1$, $M(C, \Delta) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$.

The parameter $M(C, \Delta)$

- Δ-graph: graph with maximum degree at most Δ.
- C-edge partition of G: partition of $E(G)$ into subgraphs with $\leq C$ edges.
- The problem is equivalent to determining the following parameter:

$M(C, \Delta)$: smallest integer M s.t. **any** Δ-graph has a C-edge-partition s.t. each vertex appears in $\leq M$ subgraphs.

- Therefore, we focus on determining $M(C, \Delta)$.
- W.l.o.g. we can assume that R has **regular degree** Δ.

Proposition (Lower Bound – Muñoz and S.)

For all $C, \Delta \geq 1$, $M(C, \Delta) \geq \left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$.

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

□

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\lceil \frac{\Delta}{2C} \rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

Case $\Delta \geq 2$ even

Theorem (Li and S.)

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

Proof.

- We have just seen the lower bound. Construction:
  - Orient the edges of $G = (V, E)$ in an Eulerian tour.
  - Assign to each vertex $v \in V$ its $\Delta/2$ out-edges, and partition them into $\left\lceil \frac{\Delta}{2C} \right\rceil$ stars with (at most) C edges centered at v.
  - Each vertex v appears as a leaf in stars centered at other vertices exactly $\Delta - \Delta/2 = \Delta/2$ times.
  - The number of occurrences of each vertex in this partition is

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

□

Case $\Delta \geq 3$ odd

Proposition (Upper Bound – Li and S.)

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

Corollary (Li and S.)

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

Question: is the lower bound $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$ always attained?

Theorem (Li and S.)

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

Case $\Delta \geq 3$ odd

Proposition (Upper Bound – Li and S.)

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

Corollary (Li and S.)

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

Question: is the lower bound $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$ always attained?

Theorem (Li and S.)

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

Case $\Delta \geq 3$ odd

Proposition (Upper Bound – Li and S.)

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

Corollary (Li and S.)

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

Question: is the lower bound $\left\lceil \frac{C+1}{C} \frac{\Delta}{2} \right\rceil$ always attained?

Theorem (Li and S.)

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

Case $\Delta \geq 3$ odd

Proposition (Upper Bound – Li and S.)

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

Corollary (Li and S.)

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

Question: is the lower bound $\left\lceil \frac{C+1}{C}\frac{\Delta}{2} \right\rceil$ always attained? NO!!

Theorem (Li and S.)

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

Open cases

Summarizing, we established the value of $M(C, \Delta)$ for “almost” all values of C and Δ, leaving **open** only the case where:

- $\Delta \geq 5$ is odd; and
- $C \geq 4$; and
- $3 \leq \Delta \pmod{2C} \leq C - 1$; and
- the request graph does not contain a perfect matching.

Open cases

Summarizing, we established the value of $M(C, \Delta)$ for “almost” all values of C and Δ, leaving **open** only the case where:

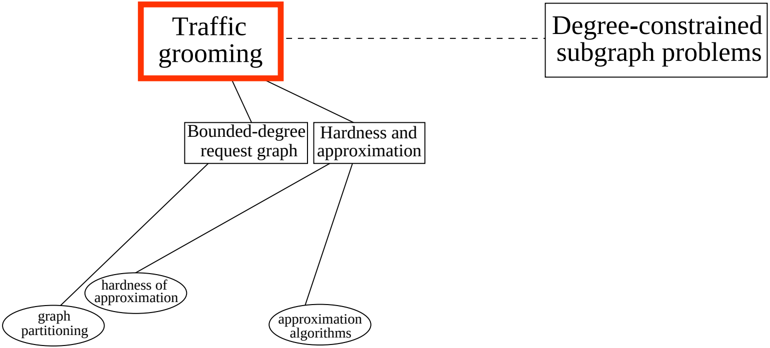
- $\Delta \geq 5$ is odd; and
- $C \geq 4$; and
- $3 \leq \Delta \pmod{2C} \leq C - 1$; and
- the request graph does not contain a perfect matching.

Open cases

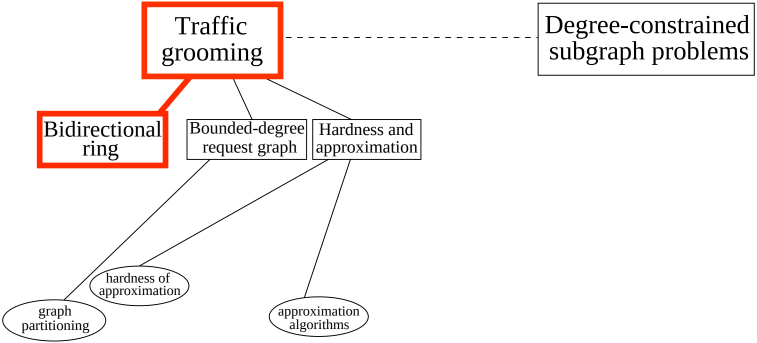
Summarizing, we established the value of $M(C, \Delta)$ for "almost" all values of C and Δ, leaving **open** only the case where:

- $\Delta \geq 5$ is odd; and
- $C \geq 4$; and
- $3 \leq \Delta \pmod{2C} \leq C - 1$; and
- the request graph does not contain a perfect matching.

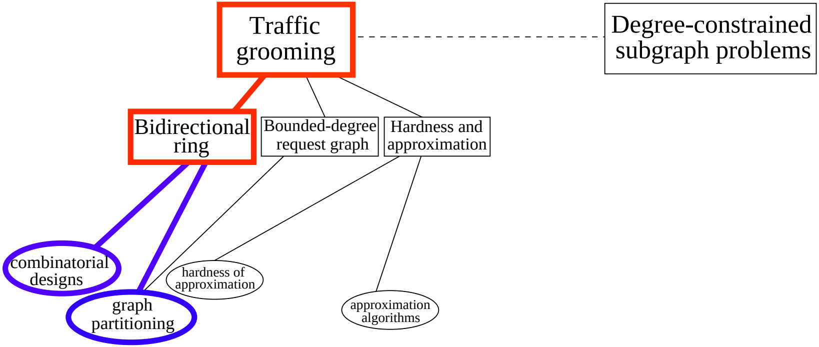
Graph of the thesis



Graph of the thesis



Graph of the thesis



Bidirectional rings

With Jean-Claude Bermond and Xavier Muñoz

- Most of the research had been done for **unidirectional rings.**
- We consider the bidirectional ring with
  - ⋆ all-to-all requests.
  - ⋆ shortest path routing.
- We provide:
  1. Statement of the problem and general lower bounds.
  2. Exhaustive study of the cases $C \in \{1, 2, 3\}$.
  3. Optimal solutions for some infinite families when $C = k(k+1)/2$.
  4. Asymptotically optimal or approximated solutions.

Bidirectional rings

With *Jean-Claude Bermond* and *Xavier Muñoz*

- Most of the research had been done for **unidirectional rings.**
- We consider the bidirectional ring with
  - ⋆ all-to-all requests.
  - ⋆ shortest path routing.
- We provide:
  1. Statement of the problem and general lower bounds.
  2. Exhaustive study of the cases $C \in \{1, 2, 3\}$.
  3. Optimal solutions for some infinite families when $C = k(k+1)/2$.
  4. Asymptotically optimal or approximated solutions.

Bidirectional rings

With *Jean-Claude Bermond* and *Xavier Muñoz*

- Most of the research had been done for **unidirectional rings.**
- We consider the bidirectional ring with
  - ⋆ all-to-all requests.
  - ⋆ shortest path routing.
- We provide:
  1. Statement of the problem and general lower bounds.
  2. Exhaustive study of the cases $C \in \{1, 2, 3\}$.
  3. Optimal solutions for some infinite families when $C = k(k+1)/2$.
  4. Asymptotically optimal or approximated solutions.

Bidirectional rings

With *Jean-Claude Bermond* and *Xavier Muñoz*

- Most of the research had been done for **unidirectional rings.**
- We consider the bidirectional ring with
  - ⋆ all-to-all requests.
  - ⋆ shortest path routing.
- We provide:
  1. Statement of the problem and general lower bounds.
  2. Exhaustive study of the cases $C \in \{1, 2, 3\}$.
  3. Optimal solutions for some infinite families when $C = k(k+1)/2$.
  4. Asymptotically optimal or approximated solutions.

Bidirectional rings

With *Jean-Claude Bermond* and *Xavier Muñoz*

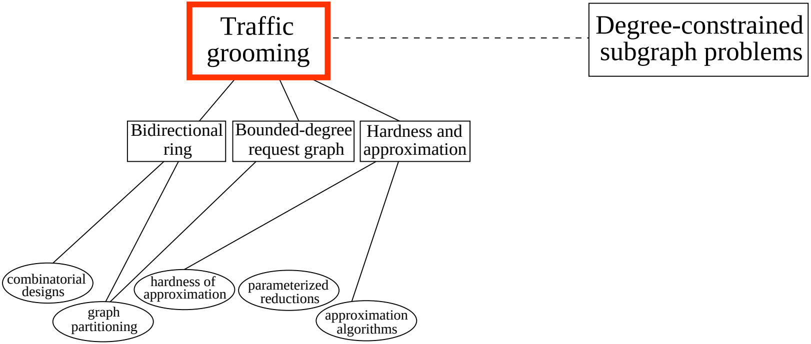
- Most of the research had been done for **unidirectional rings.**
- We consider the bidirectional ring with
  - ⋆ all-to-all requests.
  - ⋆ shortest path routing.
- We provide:
  1. Statement of the problem and general lower bounds.
  2. Exhaustive study of the cases $C \in \{1, 2, 3\}$.
  3. Optimal solutions for some infinite families when $C = k(k + 1)/2$.
  4. Asymptotically optimal or approximated solutions.

Bidirectional rings

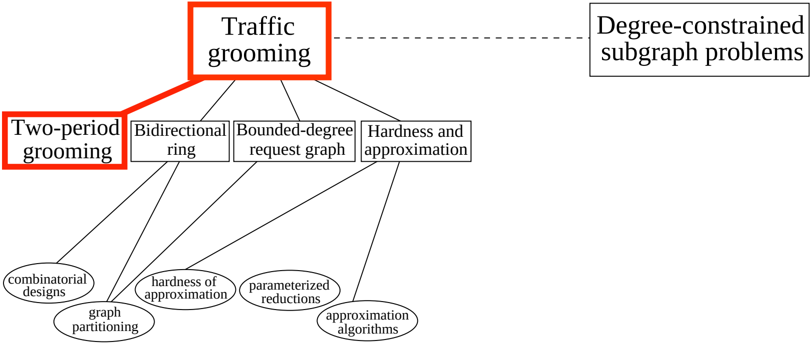
With *Jean-Claude Bermond* and *Xavier Muñoz*

- Most of the research had been done for **unidirectional rings.**
- We consider the bidirectional ring with
  - ⋆ all-to-all requests.
  - ⋆ shortest path routing.
- We provide:
  1. Statement of the problem and general lower bounds.
  2. Exhaustive study of the cases $C \in \{1, 2, 3\}$.
  3. Optimal solutions for some infinite families when $C = k(k+1)/2$.
  4. Asymptotically optimal or approximated solutions.

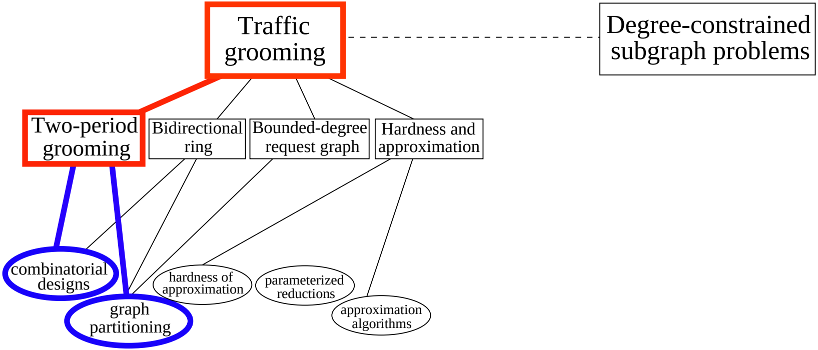
Graph of the thesis



Graph of the thesis



Graph of the thesis



2-period traffic grooming in unidirectional rings

With *J-C. Bermond*, *C.J. Colbourn*, *L. Gionfriddo*, and *G. Quattrocchi*

- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

2-period traffic grooming in unidirectional rings

With J-C. Bermond, C.J. Colbourn, L. Gionfriddo, and G. Quattrocchi

- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

2-period traffic grooming in unidirectional rings

With J-C. Bermond, C.J. Colbourn, L. Gionfriddo, and G. Quattrocchi

- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

2-period traffic grooming in unidirectional rings

With J-C. Bermond, C.J. Colbourn, L. Gionfriddo, and G. Quattrocchi

- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

2-period traffic grooming in unidirectional rings

With *J-C. Bermond, C.J. Colbourn, L. Gionfriddo*, and *G. Quattrocchi*

- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

2-period traffic grooming in unidirectional rings

With J-C. Bermond, C.J. Colbourn, L. Gionfriddo, and G. Quattrocchi

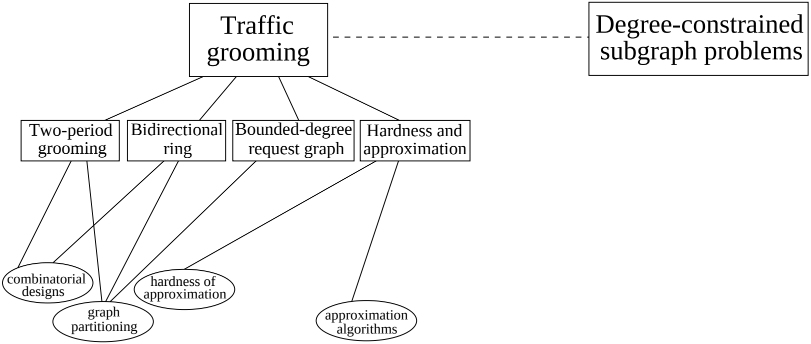
- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

2-period traffic grooming in unidirectional rings

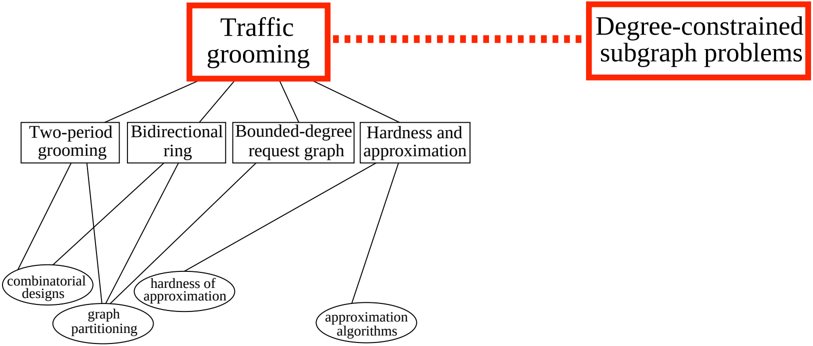
With J-C. Bermond, C.J. Colbourn, L. Gionfriddo, and G. Quattrocchi

- We consider a pseudo-dynamic scenario in unidirectional rings:
  - in the 1st period of time, there is all-to-all traffic among n nodes, each request using $1/C$ of the bandwidth.
  - in the 2nd period, there is all-to-all traffic among a subset of $n' < n$ nodes, each request using $1/C'$ of the bandwidth, with $C' < C$.
- The problem consists in finding a C-edge-partition of K_n that embeds a C'-edge-partition of $K_{n'}$.
- Introduced in [Colbourn, Quattrocchi, and Syrotiuk. *Networks'08*]. They solved the cases $C = 2$ and $C = 3$ ($C' \in \{1, 2\}$).
- We solve the case $C = 4$ (that is, $C' \in \{1, 2, 3\}$).
- In addition, we provide the optimal cost under the constraint of using the minimum number of wavelengths.

Graph of the thesis



Graph of the thesis



- Remember from the first subpart:

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3}\log^2 n)$ *for any* $C \geq 1$.

1. partition the requests into groups of **similar length [factor** $\log n$**]**
2. in each group, extract subgraphs **greedily** using an algorithm for the **DENSE k-SUBGRAPH** problem [factor $\log n$] [factor $n^{1/3}$]

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

- Summarizing, a β-approximation for the DkS problems yields a $(\beta \cdot \log^2 n)$-approximation for RING TRAFFIC GROOMING.

- Remember from the first subpart:

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3}\log^2 n)$ *for any* $C \geq 1$*.*

1. partition the requests into groups of **similar length [factor** $\log n$**]**
2. in each group, extract subgraphs **greedily** using an algorithm for the **DENSE k-SUBGRAPH** problem [factor $\log n$]   [factor $n^{1/3}$]

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

- Summarizing, a β-approximation for the DkS problems yields a $(\beta \cdot \log^2 n)$-approximation for RING TRAFFIC GROOMING.

- Remember from the first subpart:

Theorem (Amini, Pérennes, and S.)

There is a polynomial-time approximation algorithm that approximates RING TRAFFIC GROOMING *within a factor* $\mathcal{O}(n^{1/3} \log^2 n)$ *for any* $C \geq 1$*.*

1. partition the requests into groups of **similar length [factor** $\log n$**]**
2. in each group, extract subgraphs **greedily** using an algorithm for the **DENSE k-SUBGRAPH** problem [factor $\log n$] [factor $n^{1/3}$]

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

- Summarizing, a β-approximation for the DkS problems yields a $(\beta \cdot \log^2 n)$-approximation for RING TRAFFIC GROOMING.

- Remember from the first subpart:

Theorem (Amini, Pérennes, and S.)

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

1. partition the requests into groups of **similar length [factor** $\log n$**]**
2. in each group, extract subgraphs **greedily** using an algorithm for the **Dense k-subgraph** problem [factor $\log n$]   [factor $n^{1/3}$]

Dense k-Subgraph (DkS)
Input: An undirected graph $G = (V, E)$ and a positive integer k.
Output: A subset $S \subseteq V$, with $|S| = k$, such that $|E(G[S])|$ is maximized.

- Summarizing, a β-approximation for the DkS problems yields a $(\beta \cdot \log^2 n)$-approximation for Ring Traffic Grooming.

Finding dense subgraphs is difficult...

- Unfortunately, the DkS problem is a very "hard" problem:
  - Best approximation algorithm: $\mathcal{O}(n^{1/3-\varepsilon})$-approximation. [Feige, Kortsarz, and Peleg. *Algorithmica'01*]
  - Best hardness result: No PTAS, unless P=NP. [Khot. *SIAM J. Comp'06*]
- What about trying to find dense subgraphs differently?
- In DkS, the objective is to maximize the average degree
- What about the minimum degree...?

Finding dense subgraphs is difficult...

- Unfortunately, the DkS problem is a very "hard" problem:
  - Best approximation algorithm: $\mathcal{O}(n^{1/3-\varepsilon})$-approximation. [Feige, Kortsarz, and Peleg. *Algorithmica'01*]
  - Best hardness result: No PTAS, unless P$=$NP. [Khot. *SIAM J. Comp'06*]
- What about trying to find dense subgraphs differently?
- In DkS, the objective is to maximize the average degree
- What about the minimum degree...?

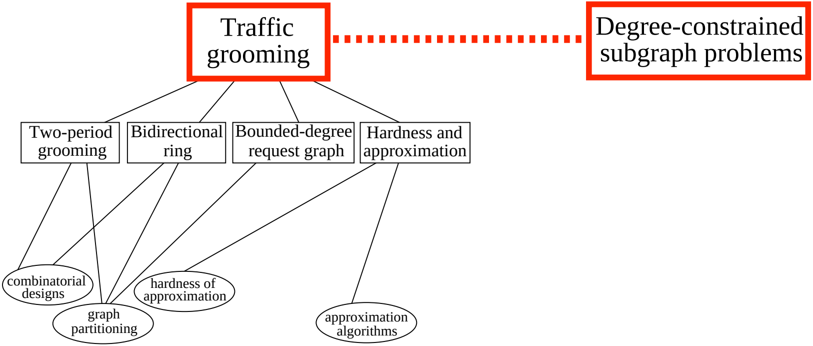
Finding dense subgraphs is difficult...

- Unfortunately, the DkS problem is a very "hard" problem:
  - Best approximation algorithm: $\mathcal{O}(n^{1/3-\varepsilon})$-approximation. [Feige, Kortsarz, and Peleg. *Algorithmica'01*]
  - Best hardness result: No PTAS, unless P$=$NP. [Khot. *SIAM J. Comp'06*]
- What about trying to find dense subgraphs differently?
- In DkS, the objective is to maximize the average degree
- What about the minimum degree...?

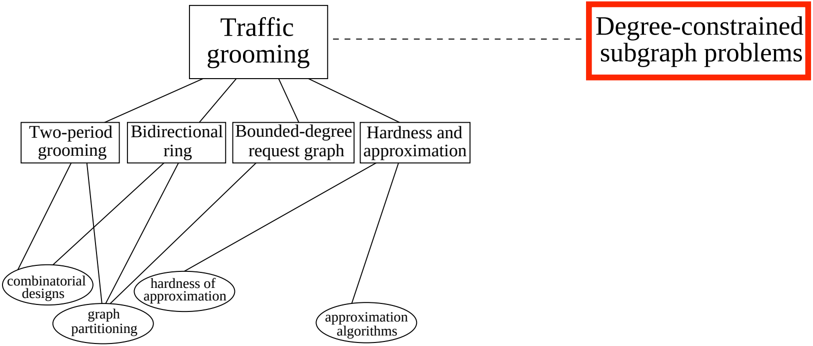
Finding dense subgraphs is difficult...

- Unfortunately, the DkS problem is a very "hard" problem:
  - Best approximation algorithm: $\mathcal{O}(n^{1/3-\varepsilon})$-approximation. [Feige, Kortsarz, and Peleg. *Algorithmica'01*]
  - Best hardness result: No PTAS, unless P$=$NP. [Khot. *SIAM J. Comp'06*]
- What about trying to find dense subgraphs differently?
- In DkS, the objective is to maximize the average degree
- What about the minimum degree...?

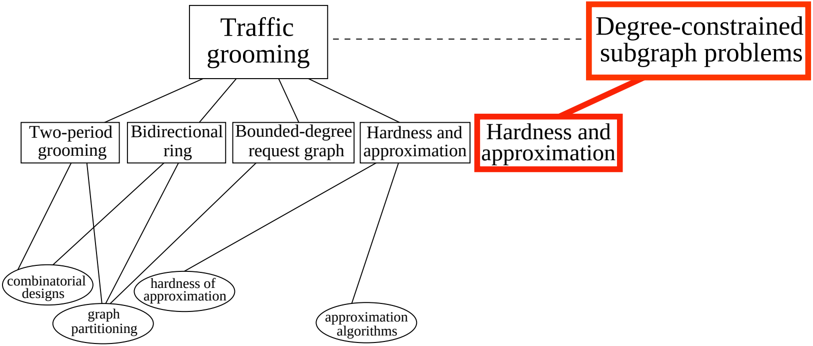
Graph of the thesis



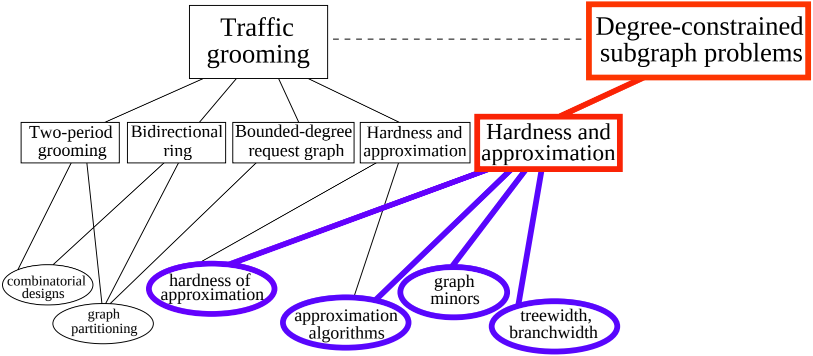
Graph of the thesis



Graph of the thesis



Graph of the thesis



Broad family of problems

A *typical* **DEGREE-CONSTRAINED SUBGRAPH PROBLEM**:

Input:

- a (*weighted* or *unweighted*) graph G, and
- an integer d.

Output:

- a (*connected*) subgraph H of G,
- satisfying some degree constraints ($\Delta(H) \leq d$ or $\delta(H) \geq d$),
- and optimizing some parameter ($|V(H)|$ or $|E(H)|$).

- Several problems in this broad family are classical widely studied NP-hard problems.
- They have a number of applications in interconnection networks, routing algorithms, chemistry, ...

Broad family of problems

A *typical* **DEGREE-CONSTRAINED SUBGRAPH PROBLEM**:

Input:

- a (*weighted* or *unweighted*) graph G, and
- an integer d.

Output:

- a (*connected*) subgraph H of G,
- satisfying some degree constraints ($\Delta(H) \leq d$ or $\delta(H) \geq d$),
- and optimizing some parameter ($|V(H)|$ or $|E(H)|$).

- Several problems in this broad family are classical widely studied NP-hard problems.
- They have a number of applications in interconnection networks, routing algorithms, chemistry, ...

Broad family of problems

A *typical* **DEGREE-CONSTRAINED SUBGRAPH PROBLEM**:

Input:

- a (*weighted* or *unweighted*) graph G, and
- an integer d.

Output:

- a (*connected*) subgraph H of G,
- satisfying some degree constraints ($\Delta(H) \leq d$ or $\delta(H) \geq d$),
- and optimizing some parameter ($|V(H)|$ or $|E(H)|$).

- Several problems in this broad family are classical widely studied NP-hard problems.
- They have a number of applications in interconnection networks, routing algorithms, chemistry, ...

Broad family of problems

A *typical* **DEGREE-CONSTRAINED SUBGRAPH PROBLEM**:

Input:

- a (*weighted* or *unweighted*) graph G, and
- an integer d.

Output:

- a (*connected*) subgraph H of G,
- satisfying some degree constraints ($\Delta(H) \leq d$ or $\delta(H) \geq d$),
- and optimizing some parameter ($|V(H)|$ or $|E(H)|$).

- Several problems in this broad family are classical widely studied NP-hard problems.
- They have a number of applications in interconnection networks, routing algorithms, chemistry, ...

First problem

MINIMUM SUBGRAPH OF MINIMUM DEGREE $\geq d$ (MSMD$_d$):

Input: an undirected graph $G = (V, E)$ and an integer $d \geq 3$.

Output: a subset $S \subseteq V$ with $\delta(G[S]) \geq d$, s.t. $|S|$ is minimum.

- For $d = 2$ it is exactly the GIRTH problem, which is in **P**.
- Therefore, it can be seen as a generalization of GIRTH.
- Is it also in **P** for $d \geq 3$?

First problem

MINIMUM SUBGRAPH OF MINIMUM DEGREE $\geq d$ (MSMD$_d$):

Input: an undirected graph $G = (V, E)$ and an integer $d \geq 3$.

Output: a subset $S \subseteq V$ with $\delta(G[S]) \geq d$, s.t. $|S|$ is minimum.

- For $d = 2$ it is exactly the GIRTH problem, which is in **P**.
- Therefore, it can be seen as a generalization of GIRTH.
- Is it also in **P** for $d \geq 3$?

First problem

MINIMUM SUBGRAPH OF MINIMUM DEGREE $\geq d$ (MSMD$_d$):

Input: an undirected graph $G = (V, E)$ and an integer $d \geq 3$.

Output: a subset $S \subseteq V$ with $\delta(G[S]) \geq d$, s.t. $|S|$ is minimum.

- For $d = 2$ it is exactly the GIRTH problem, which is in **P**.
- Therefore, it can be seen as a generalization of GIRTH.
- Is it also in **P** for $d \geq 3$?

First problem

MINIMUM SUBGRAPH OF MINIMUM DEGREE $\geq d$ (MSMD$_d$):

Input: an undirected graph $G = (V, E)$ and an integer $d \geq 3$.

Output: a subset $S \subseteq V$ with $\delta(G[S]) \geq d$, s.t. $|S|$ is minimum.

- For $d = 2$ it is exactly the GIRTH problem, which is in **P**.
- Therefore, it can be seen as a generalization of GIRTH.
- Is it also in **P** for $d \geq 3$?

Hardness and approximation

With *Omid Amini*, *David Peleg*, *Stéphane Pérennes* and *Saket Saurabh*

1. MSMD_d is not in APX for any $d \geq 3$, using the error amplification technique:
   - first we prove that MSMD_d is not in PTAS (unless P=NP).
   - then we prove that MSMD_d does not accept **any** constant factor approximation.
2. $\mathcal{O}(n/\log n)$-approximation algorithm for minor-free classes of graphs, using dynamic programming techniques and a known structural result on graph minors.

   (In particular, this applied to planar graphs and graphs of bounded genus.)

Hardness and approximation

With *Omid Amini*, *David Peleg*, *Stéphane Pérennes* and *Saket Saurabh*

1. MSMD_d is not in APX for any $d \geq 3$, using the error amplification technique:
   - first we prove that MSMD_d is not in PTAS (unless P=NP).
   - then we prove that MSMD_d does not accept **any** constant factor approximation.

2. $\mathcal{O}(n/\log n)$-approximation algorithm for minor-free classes of graphs, using dynamic programming techniques and a known structural result on graph minors.

   (In particular, this applied to planar graphs and graphs of bounded genus.)

Hardness and approximation

With Omid Amini, David Peleg, Stéphane Pérennes and Saket Saurabh

1. MSMD_d is not in APX for any $d \geq 3$, using the error amplification technique:
   - first we prove that MSMD_d is not in PTAS (unless P=NP).
   - then we prove that MSMD_d does not accept **any** constant factor approximation.
2. $\mathcal{O}(n/\log n)$-approximation algorithm for minor-free classes of graphs, using dynamic programming techniques and a known structural result on graph minors.

   (In particular, this applied to planar graphs and graphs of bounded genus.)

Second problem

MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$):

Input:

- an undirected graph $G = (V, E)$,
- an integer $d \geq 2$, and
- a weight function $\omega : E \to \mathbb{R}^+$.

Output:

a subset of edges $E' \subseteq E$ of **maximum weight**, s.t. $G' = (V, E')$

- is connected (except isolated vertices), and
- satisfies $\Delta(G') \leq d$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson, Computers and Intractability, 1979]*.
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques). [Lovász, 70's]
- For *fixed* $d = 2$ it corresponds to the LONGEST PATH problem.

Second problem

MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$):

Input:

- an undirected graph $G = (V, E)$,
- an integer $d \geq 2$, and
- a weight function $\omega : E \rightarrow \mathbb{R}^+$.

Output:

a subset of edges $E' \subseteq E$ of **maximum weight**, s.t. $G' = (V, E')$

- is connected (except isolated vertices), and
- satisfies $\Delta(G') \leq d$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson, Computers and Intractability, 1979]*.
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques). [Lovász, 70's]
- For *fixed* $d = 2$ it corresponds to the LONGEST PATH problem.

Second problem

MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$):

Input:

- an undirected graph $G = (V, E)$,
- an integer $d \geq 2$, and
- a weight function $\omega : E \to \mathbb{R}^+$.

Output:

a subset of edges $E' \subseteq E$ of **maximum weight**, s.t. $G' = (V, E')$

- is connected (except isolated vertices), and
- satisfies $\Delta(G') \leq d$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson, Computers and Intractability, 1979]*.
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques). [Lovász, 70's]
- For *fixed* $d = 2$ it corresponds to the LONGEST PATH problem.

Second problem

MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$):

Input:

- an undirected graph $G = (V, E)$,
- an integer $d \geq 2$, and
- a weight function $\omega : E \to \mathbb{R}^+$.

Output:

a subset of edges $E' \subseteq E$ of **maximum weight**, s.t. $G' = (V, E')$

- is connected (except isolated vertices), and
- satisfies $\Delta(G') \leq d$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson, Computers and Intractability, 1979]*.
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques). [Lovász, 70's]
- For *fixed* $d = 2$ it corresponds to the LONGEST PATH problem.

Second problem

MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$):

Input:

- an undirected graph $G = (V, E)$,
- an integer $d \geq 2$, and
- a weight function $\omega : E \to \mathbb{R}^+$.

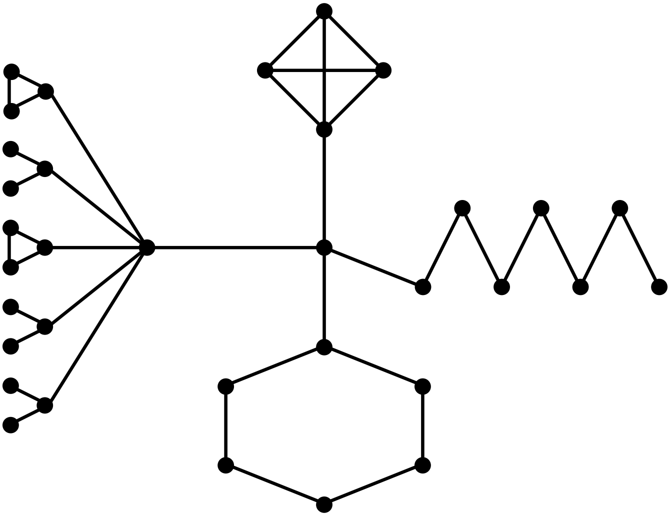
Output:

a subset of edges $E' \subseteq E$ of **maximum weight**, s.t. $G' = (V, E')$

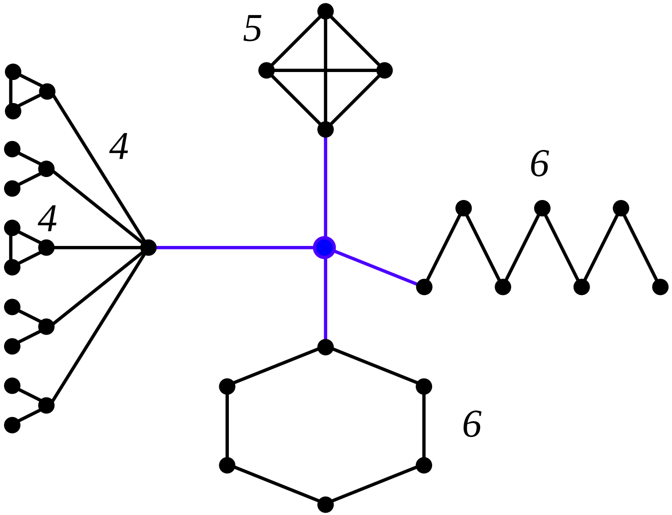
- is connected (except isolated vertices), and
- satisfies $\Delta(G') \leq d$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson, Computers and Intractability, 1979]*.
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques). [Lovász, 70's]
- For *fixed* $d = 2$ it corresponds to the LONGEST PATH problem.

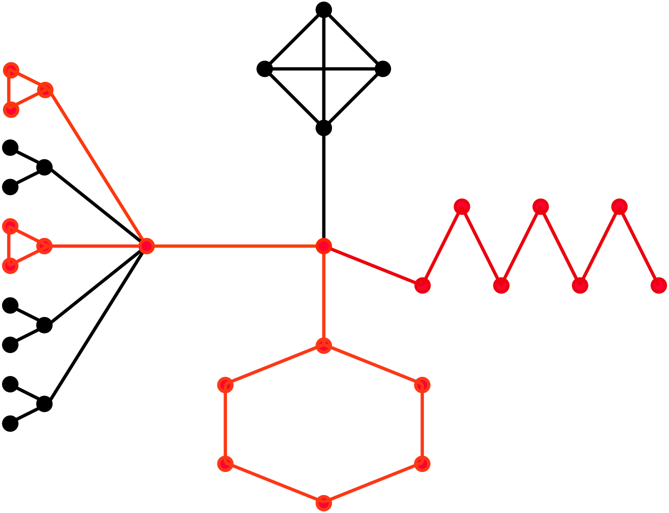
Example with $d = 3$, $\omega(e) = 1$ for all $e \in E(G)$



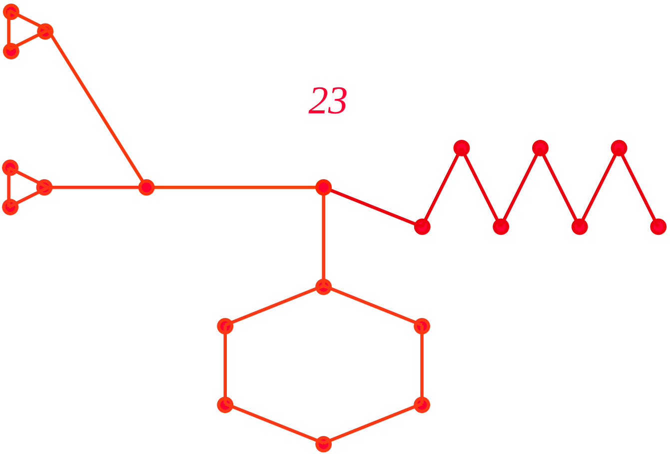
Example with $d = 3$, $\omega(e) = 1$ for all $e \in E(G)$



Example with $d = 3$, $\omega(e) = 1$ for all $e \in E(G)$



Example with $d = 3$, $\omega(e) = 1$ for all $e \in E(G)$



Hardness and approximation

With *Omid Amini*, *David Peleg*, *Stéphane Pérennes* and *Saket Saurabh*

1. not in APX for any fixed $d \geq 2$.
2. if there is a polynomial time algorithm for MDBCS_d, $d \geq 2$, with performance ratio $2^{\mathcal{O}(\sqrt{\log n})}$, then $\text{NP} \subseteq \text{DTIME}(2^{\mathcal{O}(\log^5 n)})$.
3. $\min\{m/\log n,\ nd/(2\log n)\}$-approximation algorithm for unweighted graphs. ($n = |V(G)|$ and $m = |E(G)|$)
4. $\min\{n/2,\ m/d\}$-approximation algorithm for weighted graphs.
5. if G has a low-degree spanning tree (in terms of d) it can be approximated within a small constant factor.

Hardness and approximation

With *Omid Amini, David Peleg, Stéphane Pérennes* and *Saket Saurabh*

1. not in APX for any fixed $d \geq 2$.
2. if there is a polynomial time algorithm for MDBCS_d, $d \geq 2$, with performance ratio $2^{\mathcal{O}(\sqrt{\log n})}$, then $\text{NP} \subseteq \text{DTIME}(2^{\mathcal{O}(\log^5 n)})$.
3. $\min\{m/\log n,\ nd/(2\log n)\}$-approximation algorithm for unweighted graphs. $(n = |V(G)|$ and $m = |E(G)|)$
4. $\min\{n/2,\ m/d\}$-approximation algorithm for weighted graphs.
5. if G has a low-degree spanning tree (in terms of d) it can be approximated within a small constant factor.

Hardness and approximation

With *Omid Amini*, *David Peleg*, *Stéphane Pérennes* and *Saket Saurabh*

1. not in APX for any fixed $d \geq 2$.
2. if there is a polynomial time algorithm for MDBCS_d, $d \geq 2$, with performance ratio $2^{\mathcal{O}(\sqrt{\log n})}$, then $\text{NP} \subseteq \text{DTIME}(2^{\mathcal{O}(\log^5 n)})$.
3. $\min\{m/\log n,\ nd/(2\log n)\}$-approximation algorithm for unweighted graphs. ($n = |V(G)|$ and $m = |E(G)|$)
4. $\min\{n/2,\ m/d\}$-approximation algorithm for weighted graphs.
5. if G has a low-degree spanning tree (in terms of d) it can be approximated within a small constant factor.

Hardness and approximation

With *Omid Amini, David Peleg, Stéphane Pérennes* and *Saket Saurabh*

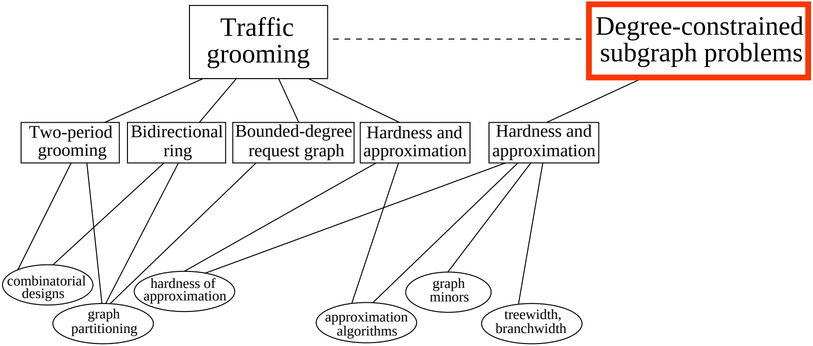
1. not in APX for any fixed $d \geq 2$.
2. if there is a polynomial time algorithm for MDBCS_d, $d \geq 2$, with performance ratio $2^{\mathcal{O}(\sqrt{\log n})}$, then $\text{NP} \subseteq \text{DTIME}(2^{\mathcal{O}(\log^5 n)})$.
3. $\min\{m/\log n,\ nd/(2\log n)\}$-approximation algorithm for unweighted graphs. ($n = |V(G)|$ and $m = |E(G)|$)
4. $\min\{n/2,\ m/d\}$-approximation algorithm for weighted graphs.
5. if G has a low-degree spanning tree (in terms of d) it can be approximated within a small constant factor.

Hardness and approximation

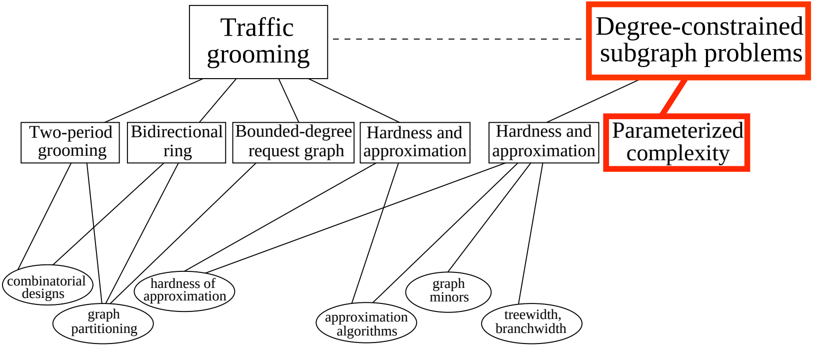
With Omid Amini, David Peleg, Stéphane Pérennes and Saket Saurabh

1. not in APX for any fixed $d \geq 2$.
2. if there is a polynomial time algorithm for MDBCS_d, $d \geq 2$, with performance ratio $2^{\mathcal{O}(\sqrt{\log n})}$, then NP $\subseteq$ DTIME$(2^{\mathcal{O}(\log^5 n)})$.
3. $\min\{m/\log n,\ nd/(2\log n)\}$-approximation algorithm for unweighted graphs. ($n = |V(G)|$ and $m = |E(G)|$)
4. $\min\{n/2,\ m/d\}$-approximation algorithm for weighted graphs.
5. if G has a low-degree spanning tree (in terms of d) it can be approximated within a small constant factor.

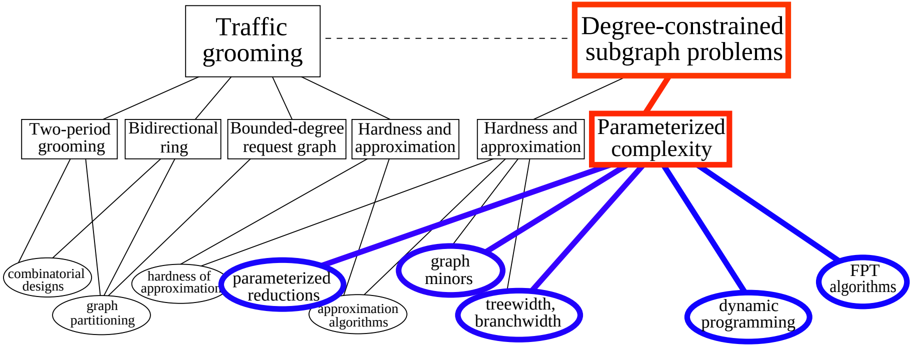
Graph of the thesis



Graph of the thesis



Graph of the thesis



Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- Barometer of intractability:

  $$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- Barometer of intractability:

  $$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more “tractable”.

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- Barometer of intractability:

  $$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- Barometer of intractability:

  $$\mathrm{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Parameterized complexity of finding degree-constrained subgraphs

With *Omid Amini* and *Saket Saurabh*

- We have studied the parameterized complexity of finding degree-constrained subgraphs, with

  parameter = number of vertices of the desired subgraph

- Namely, given two integers d and k, the problems of finding
  1. a d-regular subgraph (induced or not) with at most $\leq k$ vertices.
  2. a subgraph with at most $\leq k$ vertices and of minimum degree $\geq d$.
- We prove that
  1. these problems are $W[1]$-hard in general graphs.
  2. We then provide explicit FPT algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach.

Parameterized complexity of finding degree-constrained subgraphs

With *Omid Amini* and *Saket Saurabh*

- We have studied the parameterized complexity of finding degree-constrained subgraphs, with

  parameter $=$ number of vertices of the desired subgraph

- Namely, given two integers d and k, the problems of finding
  1. a d-regular subgraph (induced or not) with at most $\leq k$ vertices.
  2. a subgraph with at most $\leq k$ vertices and of minimum degree $\geq d$.
- We prove that
  1. these problems are $W[1]$-hard in general graphs.
  2. We then provide explicit FPT algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach.

Parameterized complexity of finding degree-constrained subgraphs

With *Omid Amini* and *Saket Saurabh*

- We have studied the parameterized complexity of finding degree-constrained subgraphs, with

  parameter = number of vertices of the desired subgraph

- Namely, given two integers d and k, the problems of finding
  1. a d-regular subgraph (induced or not) with at most $\leq k$ vertices.
  2. a subgraph with at most $\leq k$ vertices and of minimum degree $\geq d$.
- We prove that
  1. these problems are $W[1]$-hard in general graphs.
  2. We then provide explicit FPT algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach.

Parameterized complexity of finding degree-constrained subgraphs

With *Omid Amini* and *Saket Saurabh*

- We have studied the parameterized complexity of finding degree-constrained subgraphs, with

  parameter = number of vertices of the desired subgraph

- Namely, given two integers d and k, the problems of finding
  1. a d-regular subgraph (induced or not) with at most $\leq k$ vertices.
  2. a subgraph with at most $\leq k$ vertices and of minimum degree $\geq d$.
- We prove that
  1. these problems are $W[1]$-hard in general graphs.
  2. We then provide explicit FPT algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach.

Parameterized complexity of finding degree-constrained subgraphs

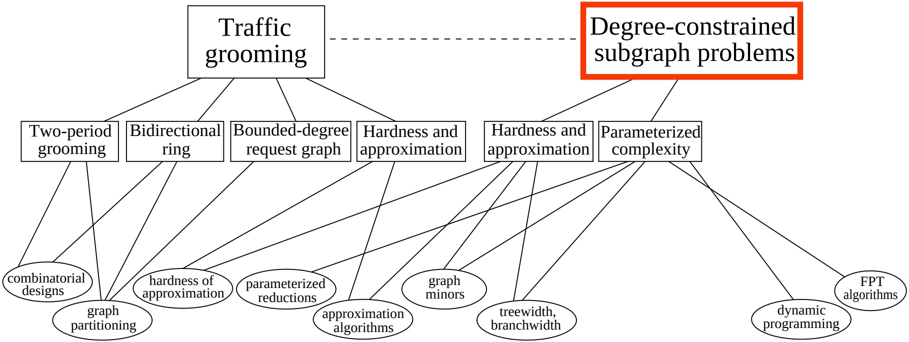
With *Omid Amini* and *Saket Saurabh*

- We have studied the parameterized complexity of finding degree-constrained subgraphs, with

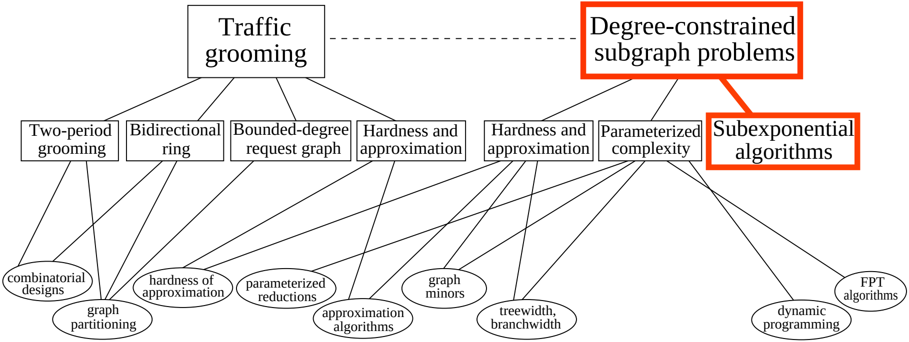
  parameter = number of vertices of the desired subgraph

- Namely, given two integers d and k, the problems of finding
  1. a d-regular subgraph (induced or not) with at most $\leq k$ vertices.
  2. a subgraph with at most $\leq k$ vertices and of minimum degree $\geq d$.
- We prove that
  1. these problems are $W[1]$-hard in general graphs.
  2. We then provide explicit FPT algorithms to solve both problems in graphs with bounded local treewidth and graphs with excluded minors, using a dynamic programming approach.

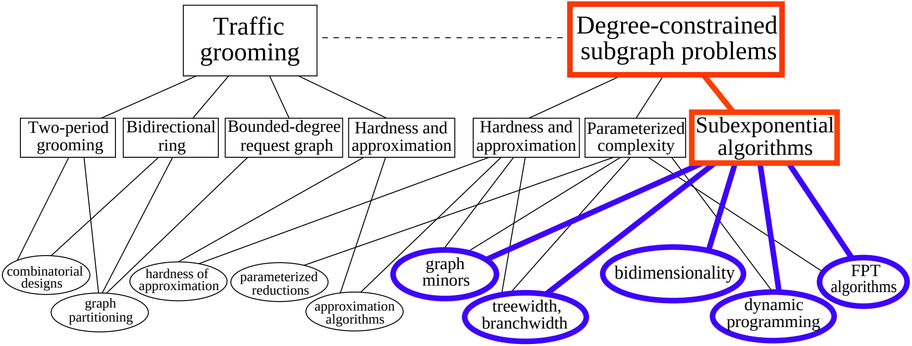
Graph of the thesis



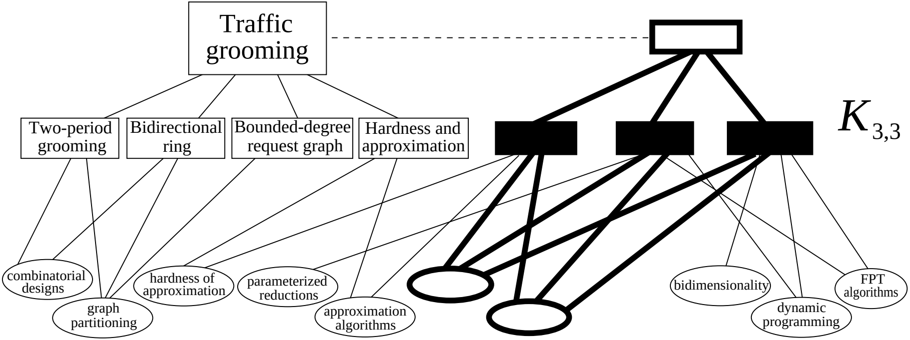
Graph of the thesis



Graph of the thesis



Graph of the thesis



FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-VERTEX COVER, k-LONGEST PATH.

- Problem: $f(k)$ can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^k}}}}$)

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-VERTEX COVER, k-LONGEST PATH.

- Problem: $f(k)$ can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^k}}}}$)

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-VERTEX COVER, k-LONGEST PATH.

- Problem: $f(k)$ can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^{k}}}}}$)
- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-Vertex Cover, k-Longest Path.

- Problem: $f(k)$ can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^{k}}}}}$)

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, **P**$(G) \leq k$?

First we compute **bw**(G). [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

bw(G) is "big" $\Rightarrow$ **P** is also "big" (typically, **P** $= \Omega(\mathbf{bw}^2)$).

- **Bidimensionality: use square grids as "certificates".** [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

(B) Dynamic programming which uses graph structure:

If **bw**(G) is "small", we decide **P** by "fast" dynamic programming.

- **Catalan structures.** [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, $\mathbf{P}(G) \leq k$?
First we compute $\mathbf{bw}(G)$. [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

$\mathbf{bw}(G)$ is "big" $\Rightarrow$ **P** is also "big" (typically, $\mathbf{P} = \Omega(\mathbf{bw}^2)$).

- **Bidimensionality: use square grids as "certificates".**
  [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

(B) Dynamic programming which uses graph structure:

If $\mathbf{bw}(G)$ is "small", we decide **P** by "fast" dynamic programming.

- **Catalan structures.**
  [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, $\mathbf{P}(G) \leq k$?
First we compute $\mathbf{bw}(G)$. [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

$\mathbf{bw}(G)$ is "big" $\Rightarrow$ **P** is also "big" (typically, $\mathbf{P} = \Omega(\mathbf{bw}^2)$).

- **Bidimensionality: use square grids as "certificates".**
  [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

(B) Dynamic programming which uses graph structure:

If $\mathbf{bw}(G)$ is "small", we decide **P** by "fast" dynamic programming.

- **Catalan structures.**
  [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, **P**$(G) \leq k$?
First we compute **bw**(G). [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

bw(G) is "big" $\Rightarrow$ **P** is also "big" (typically, **P** $= \Omega($**bw**$^2))$.

- **Bidimensionality: use square grids as "certificates".**
  [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

(B) Dynamic programming which uses graph structure:

If **bw**(G) is "small", we decide **P** by "fast" dynamic programming.

- **Catalan structures.**
  [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, $\mathbf{P}(G) \leq k$?
First we compute $\mathbf{bw}(G)$. [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

$\mathbf{bw}(G)$ is "big" $\Rightarrow$ **P** is also "big" (typically, $\mathbf{P} = \Omega(\mathbf{bw}^2)$).

- **Bidimensionality: use square grids as "certificates".** [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

(B) Dynamic programming which uses graph structure:

If $\mathbf{bw}(G)$ is "small", we decide **P** by "fast" dynamic programming.

- **Catalan structures.** [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, $\mathbf{P}(G) \leq k$?
First we compute $\mathbf{bw}(G)$. [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

$\mathbf{bw}(G)$ is "big" $\Rightarrow$ **P** is also "big" (typically, $\mathbf{P} = \Omega(\mathbf{bw}^2)$).

- **Bidimensionality: use square grids as "certificates".**
  [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

(B) Dynamic programming which uses graph structure:

If $\mathbf{bw}(G)$ is "small", we decide **P** by "fast" dynamic programming.

- **Catalan structures.**
  [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a planar graph G, $\mathbf{P}(G) \leq k$?
First we compute $\mathbf{bw}(G)$. [Seymour and Thomas. *Combinatorica'94*]

(A) Combinatorial bounds via Graph Minor theorems:

$\mathbf{bw}(G)$ is "big" $\Rightarrow$ **P** is also "big" (typically, $\mathbf{P} = \Omega(\mathbf{bw}^2)$).

- **Bidimensionality: use square grids as "certificates"**.
  [Demaine, Fomin, Hajiaghayi, Thilikos. *SODA'04, J.ACM'05*]

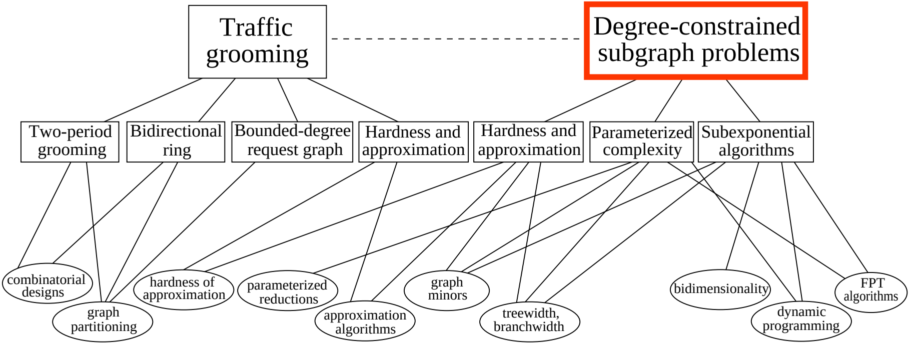
(B) Dynamic programming which uses graph structure:

If $\mathbf{bw}(G)$ is "small", we decide **P** by "fast" dynamic programming.

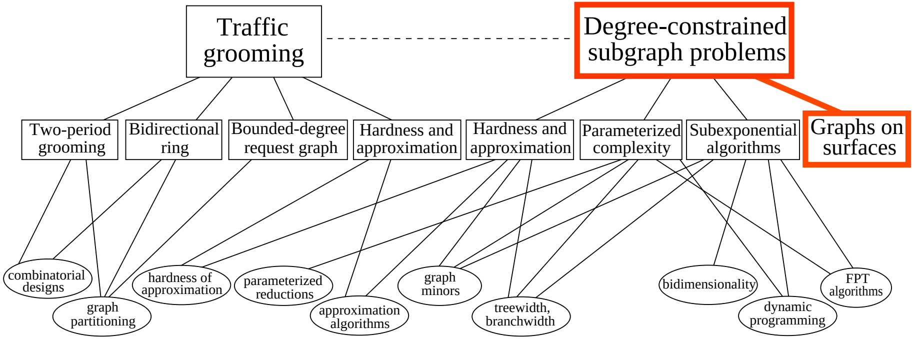
- **Catalan structures**.
  [Dorn, Fomin, Thilikos. *ICALP'07, SODA'08*]

★ With D.M. Thilikos we have adapted this framework to MDBCS_d, as well as for a few variants, introducing some general techniques.

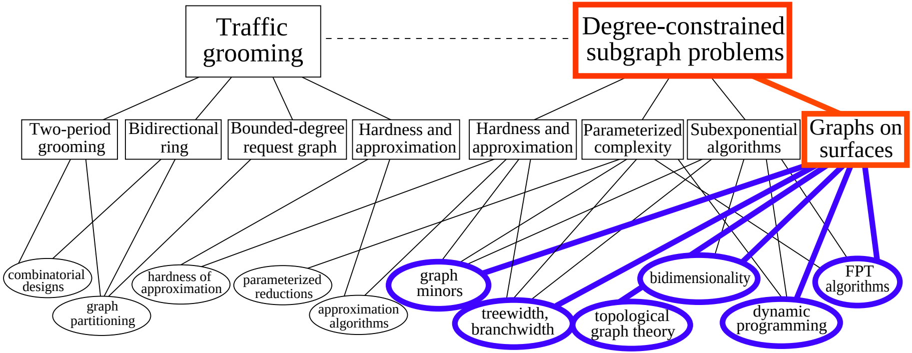
Graph of the thesis



Graph of the thesis

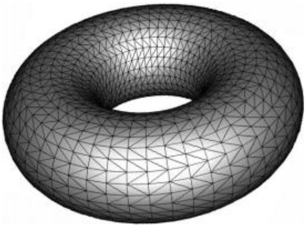


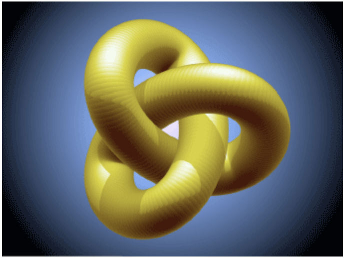
Graph of the thesis



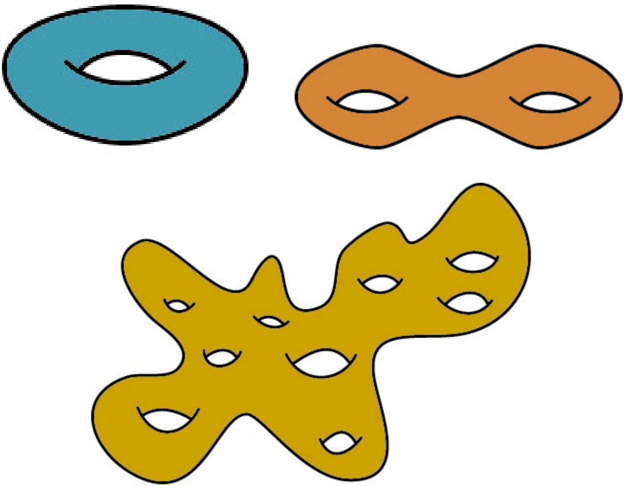
Surfaces

- Surface: connected compact 2-manifold.

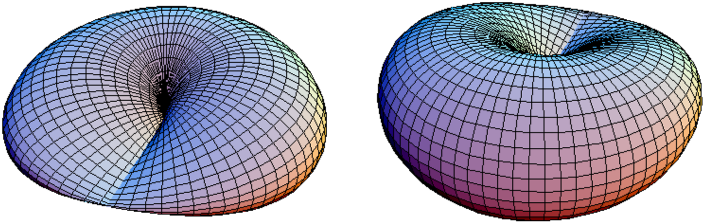




Handles



Cross-caps



Genus of a surface

- The surface classification Theorem: any compact, connected and without boundary surface can be obtained from the sphere $\mathbb{S}^2$ by adding handles and cross-caps.

- Orientable surfaces: obtained by adding $g \geq 0$ *handles* to the sphere $\mathbb{S}^2$, obtaining the g-torus $\mathbb{T}_g$ with Euler genus $\mathbf{eg}(\mathbb{T}_g) = 2g$.

- Non-orientable surfaces: obtained by adding $h > 0$ *cross-caps* to the sphere $\mathbb{S}^2$, obtaining a non-orientable surface $\mathbb{P}_h$ with Euler genus $\mathbf{eg}(\mathbb{P}_h) = h$.

Genus of a surface

- The surface classification Theorem: any compact, connected and without boundary surface can be obtained from the sphere $\mathbb{S}^2$ by adding handles and cross-caps.

- Orientable surfaces: obtained by adding $g \geq 0$ *handles* to the sphere $\mathbb{S}^2$, obtaining the g-torus $\mathbb{T}_g$ with Euler genus $\mathbf{eg}(\mathbb{T}_g) = 2g$.

- Non-orientable surfaces: obtained by adding $h > 0$ *cross-caps* to the sphere $\mathbb{S}^2$, obtaining a non-orientable surface $\mathbb{P}_h$ with Euler genus $\mathbf{eg}(\mathbb{P}_h) = h$.

Genus of a surface

- The surface classification Theorem: any compact, connected and without boundary surface can be obtained from the sphere $\mathbb{S}^2$ by adding handles and cross-caps.
- Orientable surfaces: obtained by adding $g \geq 0$ *handles* to the sphere $\mathbb{S}^2$, obtaining the g-torus $\mathbb{T}_g$ with Euler genus $\mathbf{eg}(\mathbb{T}_g) = 2g$.
- Non-orientable surfaces: obtained by adding $h > 0$ *cross-caps* to the sphere $\mathbb{S}^2$, obtaining a non-orientable surface $\mathbb{P}_h$ with Euler genus $\mathbf{eg}(\mathbb{P}_h) = h$.

Graphs on surfaces

- An embedding of a graph G on a surface Σ is a drawing of G on Σ without edge crossings.
- An embedding defines vertices, edges, and faces.
- The Euler genus of a graph G, $\mathbf{eg}(G)$, is the least Euler genus of the surfaces in which G can be embedded.

Graphs on surfaces

- An embedding of a graph G on a surface Σ is a drawing of G on Σ without edge crossings.
- An embedding defines vertices, edges, and faces.
- The Euler genus of a graph G, $\mathbf{eg}(G)$, is the least Euler genus of the surfaces in which G can be embedded.

Dynamic programming for graphs on surfaces

With *Juanjo Rué* and *Dimitrios M. Thilikos*

- Let G be a graph on n vertices with branchwidth at most k.
- We consider graph problems for which dynamic programming uses tables encoding vertex partitions ("Category (C)").

  For instance, our approach applies to MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH, MAXIMUM d-DEGREE-BOUNDED CONNECTED INDUCED SUBGRAPH and several variants, CONNECTED DOMINATING SET, CONNECTED r-DOMINATION, CONNECTED FVS, MAXIMUM LEAF SPANNING TREE, MAXIMUM FULL-DEGREE SPANNING TREE, MAXIMUM EULERIAN SUBGRAPH, STEINER TREE, MAXIMUM LEAF TREE, ...
- For general graphs, the best known algorithms for such problems run in $k^{O(k)} \cdot n$ steps.

Dynamic programming for graphs on surfaces

With *Juanjo Rué* and *Dimitrios M. Thilikos*

- Let G be a graph on n vertices with branchwidth at most k.
- We consider graph problems for which dynamic programming uses tables encoding vertex partitions ("Category (C)").

  For instance, our approach applies to MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH, MAXIMUM d-DEGREE-BOUNDED CONNECTED INDUCED SUBGRAPH and several variants, CONNECTED DOMINATING SET, CONNECTED r-DOMINATION, CONNECTED FVS, MAXIMUM LEAF SPANNING TREE, MAXIMUM FULL-DEGREE SPANNING TREE, MAXIMUM EULERIAN SUBGRAPH, STEINER TREE, MAXIMUM LEAF TREE, ...

- For general graphs, the best known algorithms for such problems run in $k^{\mathcal{O}(k)} \cdot n$ steps.

Dynamic programming for graphs on surfaces

With *Juanjo Rué* and *Dimitrios M. Thilikos*

- Let G be a graph on n vertices with branchwidth at most k.
- We consider graph problems for which dynamic programming uses tables encoding vertex partitions ("Category (C)").

  For instance, our approach applies to MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH, MAXIMUM d-DEGREE-BOUNDED CONNECTED INDUCED SUBGRAPH and several variants, CONNECTED DOMINATING SET, CONNECTED r-DOMINATION, CONNECTED FVS, MAXIMUM LEAF SPANNING TREE, MAXIMUM FULL-DEGREE SPANNING TREE, MAXIMUM EULERIAN SUBGRAPH, STEINER TREE, MAXIMUM LEAF TREE, ...
- For general graphs, the best known algorithms for such problems run in $k^{\mathcal{O}(k)} \cdot n$ steps.

Dynamic programming for graphs on surfaces

With *Juanjo Rué* and *Dimitrios M. Thilikos*

- Let G be a graph on n vertices with branchwidth at most k.
- We consider graph problems for which dynamic programming uses tables encoding vertex partitions ("Category (C)").

  For instance, our approach applies to MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH, MAXIMUM d-DEGREE-BOUNDED CONNECTED INDUCED SUBGRAPH and several variants, CONNECTED DOMINATING SET, CONNECTED r-DOMINATION, CONNECTED FVS, MAXIMUM LEAF SPANNING TREE, MAXIMUM FULL-DEGREE SPANNING TREE, MAXIMUM EULERIAN SUBGRAPH, STEINER TREE, MAXIMUM LEAF TREE, . . .

- For general graphs, the best known algorithms for such problems run in $k^{\mathcal{O}(k)} \cdot n$ steps.

From sphere to surface cut decompositions

- We build a framework for the design of $2^{\mathcal{O}(k)} \cdot n$ step dynamic programming algorithms on **surface-embedded** graphs.
- In particular, our results imply and improve all the results in [Dorn, Fomin, and Thilikos. *SWAT'06*]
- Our approach is based on a new type of branch decomposition, called surface cut decomposition.
- Surface cut decompositions for graphs on surfaces **generalize** sphere cut decompositions for planar graphs. [Seymour and Thomas. *Combinatorica'94*]

From sphere to surface cut decompositions

- We build a framework for the design of $2^{\mathcal{O}(k)} \cdot n$ step dynamic programming algorithms on **surface-embedded** graphs.
- In particular, our results imply and improve all the results in [Dorn, Fomin, and Thilikos. *SWAT'06*]
- Our approach is based on a new type of branch decomposition, called surface cut decomposition.
- Surface cut decompositions for graphs on surfaces **generalize** sphere cut decompositions for planar graphs. [Seymour and Thomas. *Combinatorica'94*]

From sphere to surface cut decompositions

- We build a framework for the design of $2^{\mathcal{O}(k)} \cdot n$ step dynamic programming algorithms on **surface-embedded** graphs.
- In particular, our results imply and improve all the results in [Dorn, Fomin, and Thilikos. *SWAT'06*]
- Our approach is based on a new type of branch decomposition, called surface cut decomposition.
- Surface cut decompositions for graphs on surfaces **generalize** sphere cut decompositions for planar graphs. [Seymour and Thomas. *Combinatorica'94*]

From sphere to surface cut decompositions

- We build a framework for the design of $2^{\mathcal{O}(k)} \cdot n$ step dynamic programming algorithms on **surface-embedded** graphs.
- In particular, our results imply and improve all the results in [Dorn, Fomin, and Thilikos. *SWAT'06*]
- Our approach is based on a new type of branch decomposition, called surface cut decomposition.
- Surface cut decompositions for graphs on surfaces **generalize** sphere cut decompositions for planar graphs. [Seymour and Thomas. *Combinatorica'94*]

Nooses

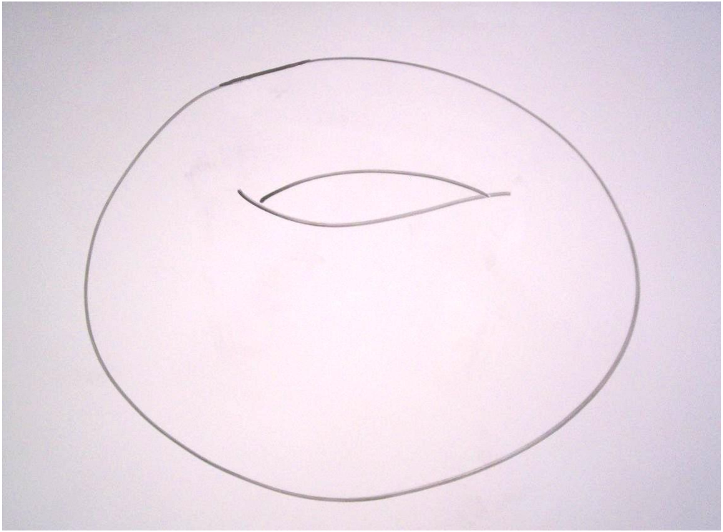
- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.

Nooses

- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.

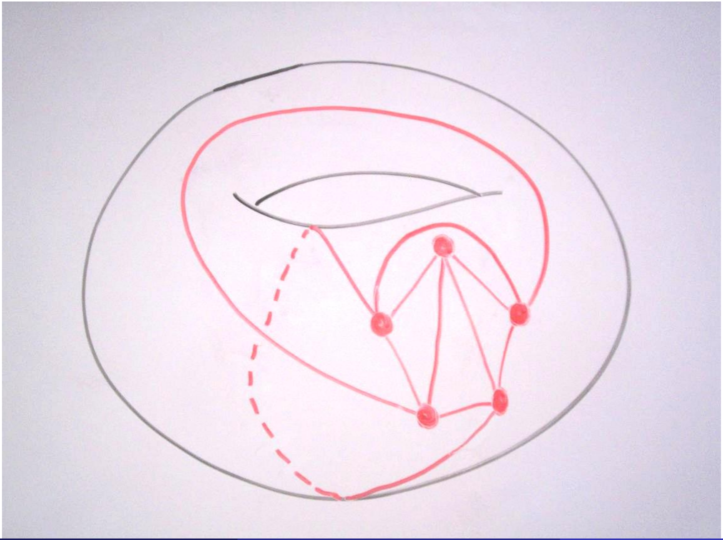
Nooses

- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.



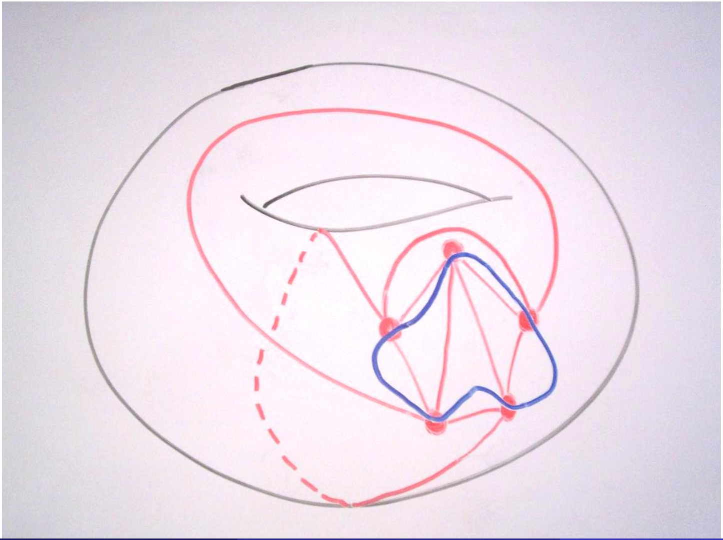
Nooses

- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.



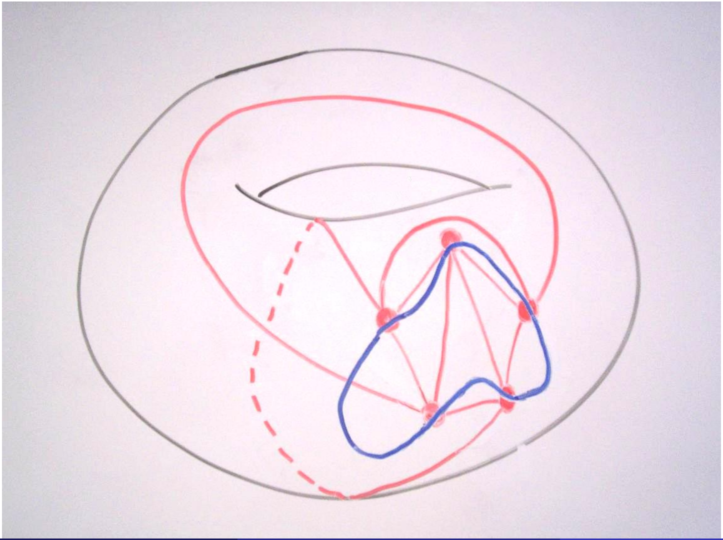
Nooses

- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.



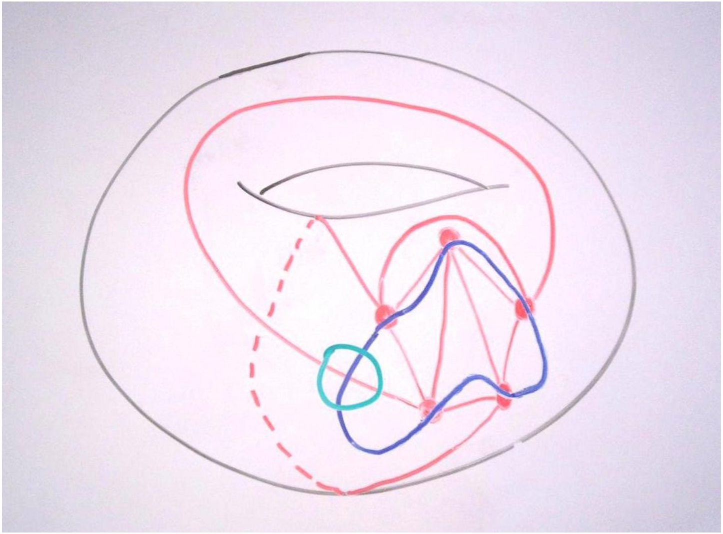
Nooses

- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.



Nooses

- Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.



Sphere cut decompositions

- *Sphere cut decomposition*: Branch decomposition where the vertices in each **mid**(e) are situated around a noose.
- The size of the tables of a dynamic programming algorithm depend on how many ways a partial solution can intersect **mid**(e).
- In how many ways we can draw polygons inside a circle such that they touch the circle only on its vertices and they do not intersect?
- Exactly the number of *non-crossing partitions* over ℓ elements, which is given by the ℓ-th Catalan number:

$$\text{CN}(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell.$$

Sphere cut decompositions

- *Sphere cut decomposition*: Branch decomposition where the vertices in each **mid**(e) are situated around a noose.
- The size of the tables of a dynamic programming algorithm depend on how many ways a partial solution can intersect **mid**(e).
- In how many ways we can draw polygons inside a circle such that they touch the circle only on its vertices and they do not intersect?
- Exactly the number of *non-crossing partitions* over ℓ elements, which is given by the ℓ-th Catalan number:

$$\text{CN}(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell.$$

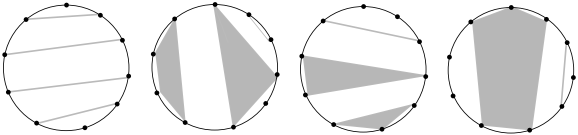
Sphere cut decompositions

- *Sphere cut decomposition*: Branch decomposition where the vertices in each **mid**(e) are situated around a noose.
- The size of the tables of a dynamic programming algorithm depend on how many ways a partial solution can intersect **mid**(e).
- In how many ways we can draw polygons inside a circle such that they touch the circle only on its vertices and they do not intersect?
- Exactly the number of *non-crossing partitions* over ℓ elements, which is given by the ℓ-th Catalan number:

$$\text{CN}(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell.$$

Sphere cut decompositions

- *Sphere cut decomposition*: Branch decomposition where the vertices in each **mid**(e) are situated around a noose.
- The size of the tables of a dynamic programming algorithm depend on how many ways a partial solution can intersect **mid**(e).
- In how many ways we can draw polygons inside a circle such that they touch the circle only on its vertices and they do not intersect?



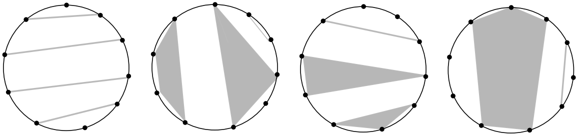
- Exactly the number of *non-crossing partitions* over ℓ elements, which is given by the ℓ-th Catalan number:

$$\text{CN}(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell.$$

Sphere cut decompositions

- *Sphere cut decomposition*: Branch decomposition where the vertices in each **mid**(e) are situated around a noose.

- The size of the tables of a dynamic programming algorithm depend on how many ways a partial solution can intersect **mid**(e).
- In how many ways we can draw polygons inside a circle such that they touch the circle only on its vertices and they do not intersect?



- Exactly the number of *non-crossing partitions* over ℓ elements, which is given by the ℓ-th Catalan number:

$$\text{CN}(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell.$$

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

A surface cut decomposition of G is a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\mathbf{g})$, s.t. for all $e \in E(T)$

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - ⋆ the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\mathbf{g})$ nooses;
  - ⋆ these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
  - ⋆ $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

A surface cut decomposition of G is a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\mathbf{g})$, s.t. for all $e \in E(T)$

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - ⋆ the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\mathbf{g})$ nooses;
  - ⋆ these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
  - ⋆ $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

A surface cut decomposition of G is a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\mathbf{g})$, s.t. for all $e \in E(T)$

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - ⋆ the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\mathbf{g})$ nooses;
  - ⋆ these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
  - ⋆ $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

A surface cut decomposition of G is a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\mathbf{g})$, s.t. for all $e \in E(T)$

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - ⋆ the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\mathbf{g})$ nooses;
  - ⋆ these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
  - ⋆ $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

A surface cut decomposition of G is a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\mathbf{g})$, s.t. for all $e \in E(T)$

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - ⋆ the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\mathbf{g})$ nooses;
  - ⋆ these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
  - ⋆ $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

A surface cut decomposition of G is a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\mathbf{g})$, s.t. for all $e \in E(T)$

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - ⋆ the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\mathbf{g})$ nooses;
  - ⋆ these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
  - ⋆ $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

How to use surface cut decompositions?

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, one can construct in* $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ *time a surface cut decomposition* (T, μ) *of G of width at most* $27k + \mathcal{O}(\mathbf{g})$*.*

The main result is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is single exponential:

Theorem (Rué, Thilikos, and S.)

Given a problem P belonging to Category (C) in a graph G embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by* $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\mathbf{g})} \cdot \mathbf{g}^{\mathcal{O}(\mathbf{g})}$*.*

This fact is proved using topological graph theory and analytic combinatorics, generalizing Catalan structures to arbitrary surfaces.

How to use surface cut decompositions?

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$, *with* $\mathbf{bw}(G) \leq k$, *one can construct in* $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ *time a surface cut decomposition* (T, μ) *of G of width at most* $27k + \mathcal{O}(\mathbf{g})$.

The main result is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is single exponential:

Theorem (Rué, Thilikos, and S.)

Given a problem P belonging to Category (C) in a graph G embedded in a surface of Euler genus $\mathbf{g}$, *with* $\mathbf{bw}(G) \leq k$, *the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by* $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\mathbf{g})} \cdot \mathbf{g}^{\mathcal{O}(\mathbf{g})}$.

This fact is proved using topological graph theory and analytic combinatorics, generalizing Catalan structures to arbitrary surfaces.

How to use surface cut decompositions?

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$, with $\mathbf{bw}(G) \leq k$, one can construct in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\mathbf{g})$.

The main result is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is single exponential:

Theorem (Rué, Thilikos, and S.)

Given a problem P belonging to Category (C) in a graph G embedded in a surface of Euler genus $\mathbf{g}$, with $\mathbf{bw}(G) \leq k$, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\mathbf{g})} \cdot \mathbf{g}^{\mathcal{O}(\mathbf{g})}$.

This fact is proved using topological graph theory and analytic combinatorics, generalizing Catalan structures to arbitrary surfaces.

How to use surface cut decompositions?

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, one can construct in* $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ *time a surface cut decomposition* (T, μ) *of G of width at most* $27k + \mathcal{O}(\mathbf{g})$*.*

The main result is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is single exponential:

Theorem (Rué, Thilikos, and S.)

Given a problem P belonging to Category (C) in a graph G embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by* $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\mathbf{g})} \cdot \mathbf{g}^{\mathcal{O}(\mathbf{g})}$*.*

This fact is proved using topological graph theory and analytic combinatorics, generalizing Catalan structures to arbitrary surfaces.

How to use surface cut decompositions?

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, one can construct in* $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ *time a surface cut decomposition* (T, μ) *of G of width at most* $27k + \mathcal{O}(\mathbf{g})$*.*

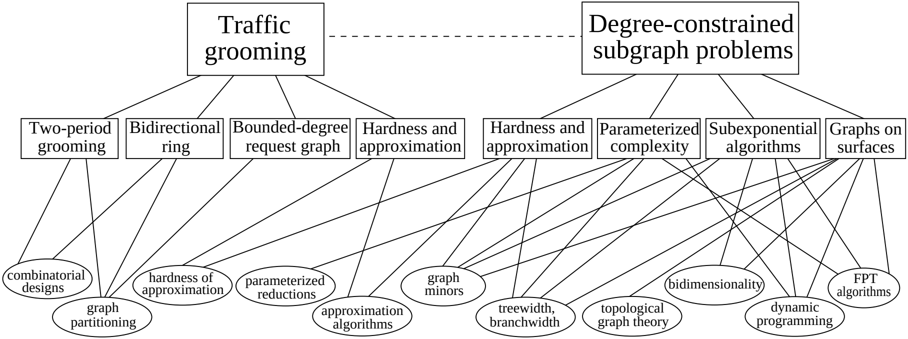
The main result is that if dynamic programming is applied on surface cut decompositions, then the time dependence on branchwidth is single exponential:

Theorem (Rué, Thilikos, and S.)

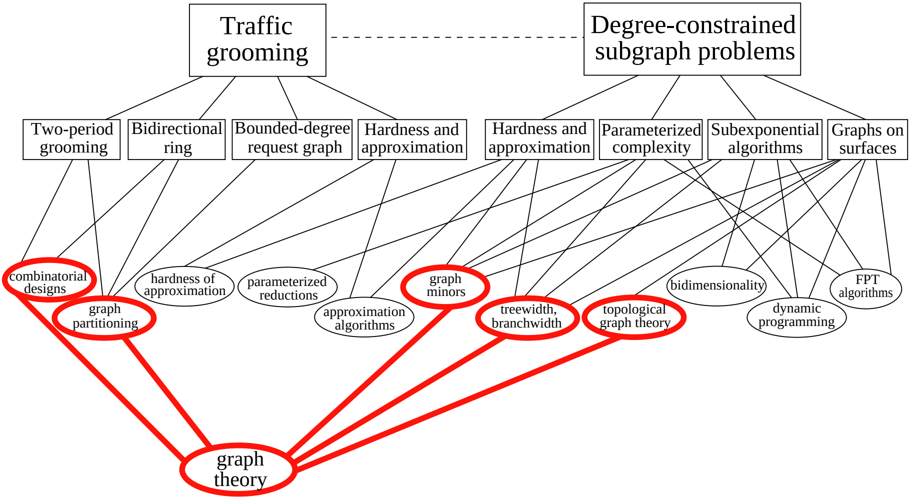
Given a problem P belonging to Category (C) in a graph G embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by* $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\mathbf{g})} \cdot \mathbf{g}^{\mathcal{O}(\mathbf{g})}$*.*

This fact is proved using topological graph theory and analytic combinatorics, generalizing Catalan structures to arbitrary surfaces.

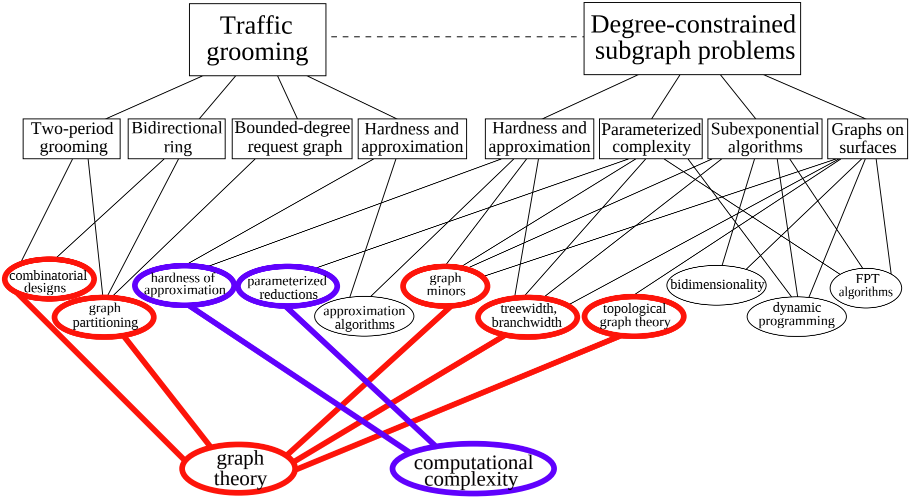
Graph of the thesis



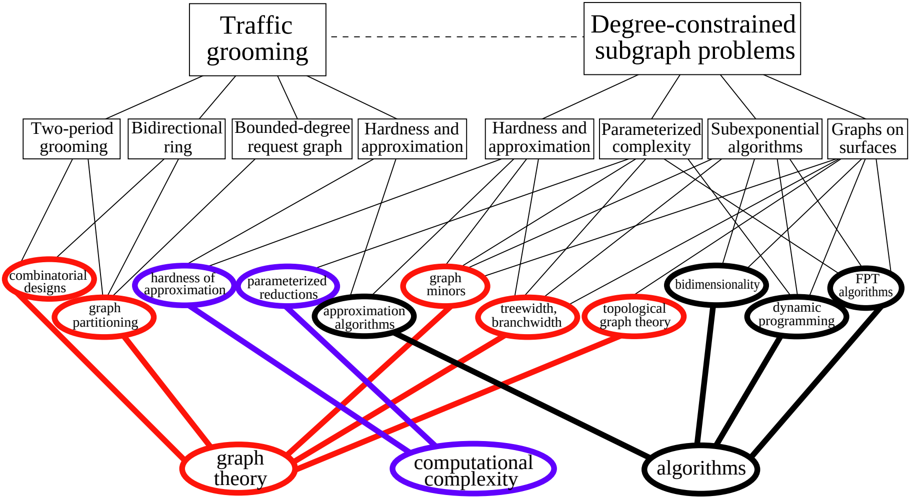
Graph of the thesis



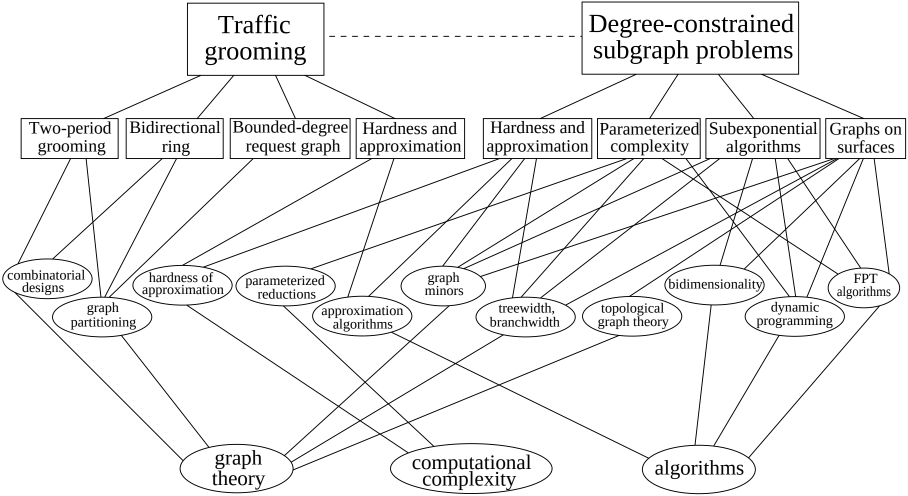
Graph of the thesis



Graph of the thesis



Graph of the thesis



Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, . . .

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, . . .

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, . . .

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, . . .

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, ...

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, . . .

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, ...

Further research

- Open problems and conjectures in each chapter of the manuscript.
- Traffic grooming:
  - Close the complexity gap when C is part of the input.
  - In rings, determine the best routing for each request graph.
  - Consider other physical topologies.
- Where is the limit of generalization? algorithmic meta-theorems
- Better understand the structure and the algorithmic properties of sparse families of graphs.
- Graph coloring, probabilistic method, . . .

Gràcies!