# Hardness and Approximation of Traffic Grooming

*Omid Amini* *Stéphane Pérennes* **Ignasi Sau**

Projet MASCOTTE, CNRS/I3S-INRIA-UNSA, France

COST 293 - Rome

# Outline of the talk

- Introduction
- Traffic Grooming
  - Definition of the problem
  - Example
  - State of the art
  - Evolution in COST 293
- Problem of MECT-B
- Hardness results
- Approximation algorithm
- Conclusions

# How to prove hardness results

- **Class APX (Approximable):**

  an optimization problem is in APX if it admits to be approximated within a constant factor.

  ***Example***: VERTEX COVER

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

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

  ***Ex.***: TRAVELING SALESMAN PROBLEM *in the Euclidean plane*

- We know that

$$\text{PTAS} \subsetneq \text{APX}$$

- So, if $\Pi$ is an optimization problem:

$$\Pi \text{ is APX-hard} \quad \Rightarrow \quad \Pi \notin \text{PTAS}$$

# How to prove hardness results

- **Class APX (Approximable):**

  an optimization problem is in APX if it admits to be approximated within a constant factor.

  ***Example:*** VERTEX COVER

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

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

  ***Ex.:*** TRAVELING SALESMAN PROBLEM *in the Euclidean plane*

- We know that

$$\text{PTAS} \subsetneq \text{APX}$$

- So, if $\Pi$ is an optimization problem:

$$\Pi \text{ is APX-hard} \quad \Rightarrow \quad \Pi \notin \text{PTAS}$$

# How to prove hardness results

- **Class APX (Approximable):**

  an optimization problem is in APX if it admits to be approximated within a constant factor.

  ***Example**:* VERTEX COVER

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

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

  ***Ex.**:* TRAVELING SALESMAN PROBLEM *in the Euclidean plane*

- We know that

$$\text{PTAS} \subsetneq \text{APX}$$

- So, if $\Pi$ is an optimization problem:

$$\Pi \text{ is APX-hard} \quad \Rightarrow \quad \Pi \notin \text{PTAS}$$

# Traffic Grooming

# Introduction

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

  **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)

# Introduction

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

# Introduction

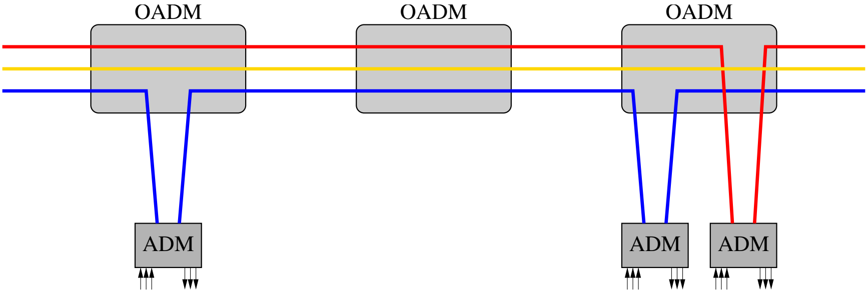
- WDM (Wavelength Division Multiplexing) networks
  - 1 wavelength (or frequency) = up to 40 Gb/s
  - 1 fiber = hundreds of wavelengths = Tb/s
- Idea

  **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)

# ADM and OADM

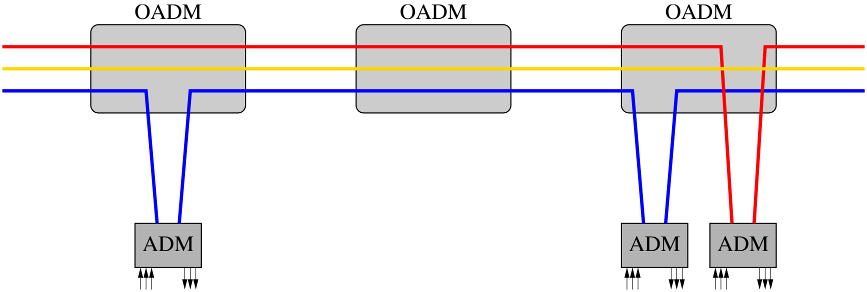
- **OADM** (Optical Add/Drop Multiplexer)= insert/extract a wavelength to/from an optical fiber
- **ADM** (Add/Drop Multiplexer)= insert/extract an OC/STM (electric low-speed signal) to/from a wavelength




⟶ we want to minimize the number of ADMs

# 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



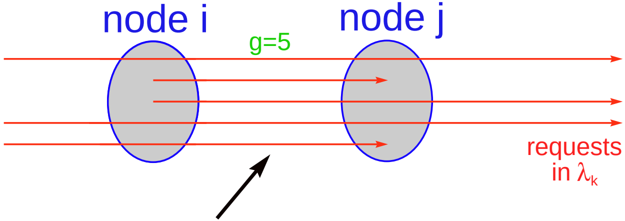

⟶ we want to minimize the number of ADMs

# Definitions

- **Request** $(i, j)$: a pair of vertices $i, j$ that want to exchange (low-speed) traffic

# Definitions

- **Request** $(i, j)$: a pair of vertices $(i, j)$ that want to exchange (low-speed) traffic
- **Grooming factor** $g$**:**

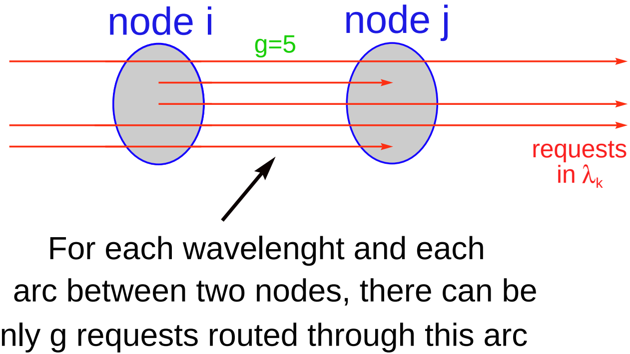



For each wavelenght and each
arc between two nodes, there can be
only g requests routed through this arc

Example:
Capacity of one wavelength $= 2400\ Mb/s$
Capacity used by a request $= 600\ Mb/s$ $\quad \Rightarrow \quad g = 4$

# Definitions

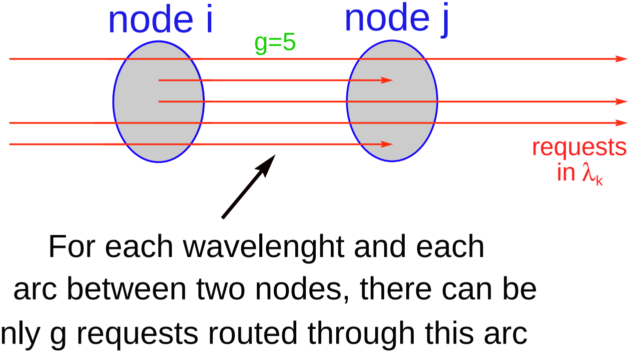
- **Request** $(i, j)$: a pair of vertices $(i, j)$ that want to exchange (low-speed) traffic
- **Grooming factor** $g$**:**




Example:
Capacity of one wavelength $= 2400\ Mb/s$
Capacity used by a request $= 600\ Mb/s$ $\Rightarrow\ g = 4$

# Definitions

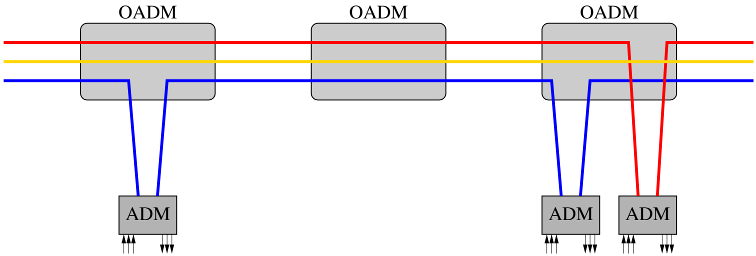
- **Request** $(i, j)$: a pair of vertices $(i, j)$ that want to exchange (low-speed) traffic
- **Grooming factor** $g$**:**




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

# 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




- <u>**Idea:**</u> 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 | $\rightarrow$ | oriented graph $G$ |
| Request set | $\rightarrow$ | oriented graph $R$ |
| Grooming factor | $\rightarrow$ | integer $g$ |
| Request in a wavelength | $\rightarrow$ | arcs in a subgraph of $R$ |
| ADM in a wavelength | $\rightarrow$ | node in a subgraph of $R$ |

- We study the cases when $G = C_n$ (ring) or $G = P_n$ (path)

# To fix ideas...

- Model:

| | | |
|---|---|---|
| Topology | $\rightarrow$ | oriented graph $G$ |
| Request set | $\rightarrow$ | oriented graph $R$ |
| Grooming factor | $\rightarrow$ | integer $g$ |
| Request in a wavelength | $\rightarrow$ | arcs in a subgraph of $R$ |
| ADM in a wavelength | $\rightarrow$ | node in a subgraph of $R$ |

- We study the cases when $G = C_n$ (ring) or $G = P_n$ (path)

# Statement of the problem

## Ring traffic grooming

**Input**

A cycle $C_n$ on $n$ nodes (network)
An oriented graph $R$ (request set)
A grooming factor $g$

**Output**

Find for each arc $r \in R$ a path $P(r)$ in $C_n$, and a partition of the arcs of $R$ into subgraphs $R_\omega$, $1 \leq \omega \leq W$, in such a way that

$$\forall\, e \in E(C_n), \qquad \text{load}(R_\omega, e) \leq g$$

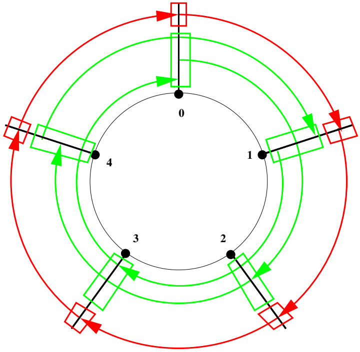
**Objective**

Minimize $\sum_{\omega=1}^{W} |V(R_\omega)|$

# Example: $n = 5$, $R = K_5$, and $g = 2$

# Example: $n = 5$, $R = K_5$, and $g = 2$

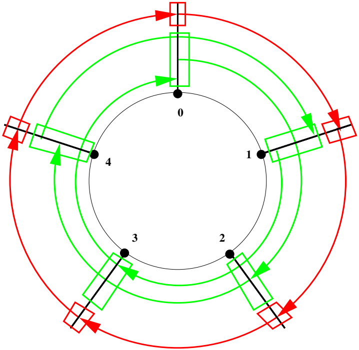
We partition the edges of $R$ in two ways, both using two wavelengths:



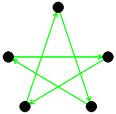

**10 ADMS**

# Example: $n = 5$, $R = K_5$, and $g = 2$

We partition the edges of $R$ in two ways, both using two wavelengths:

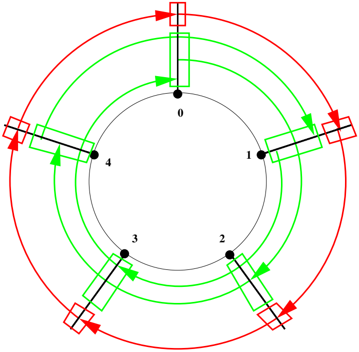



**10 ADMS**

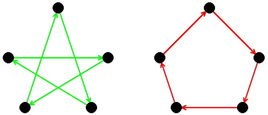


# Example: $n = 5$, $R = K_5$, and $g = 2$

We partition the edges of $R$ in two ways, both using two wavelengths:

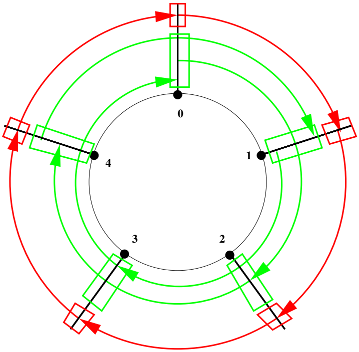



**10 ADMS**

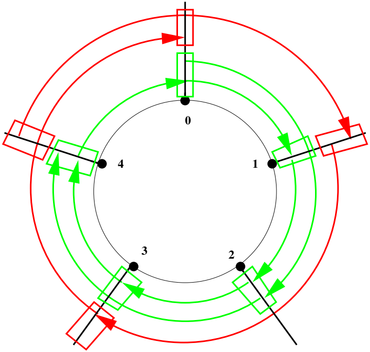


# Example: $n = 5$, $R = K_5$, and $g = 2$

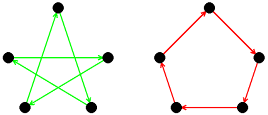
We partition the edges of $R$ in two ways, both using two wavelengths:




**10 ADMS**

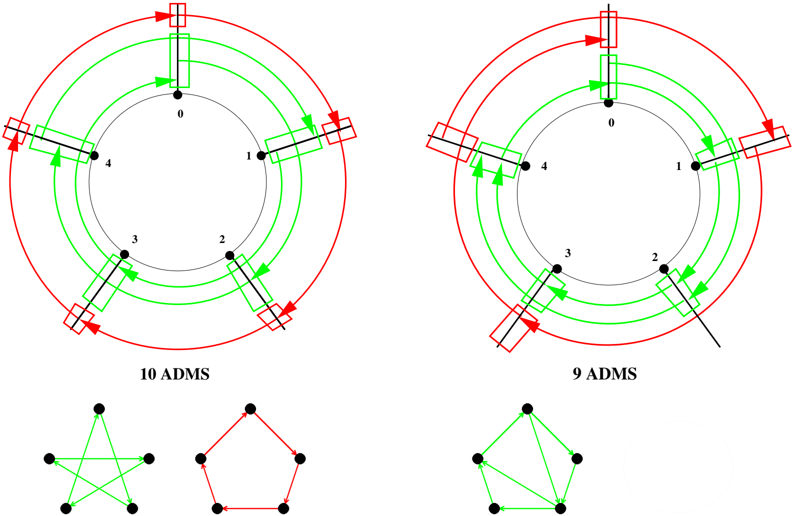



**9 ADMS**



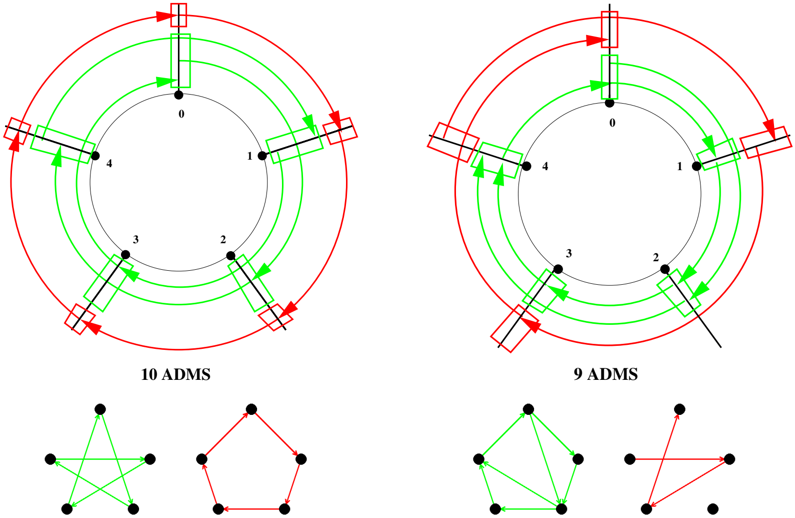
# Example: $n = 5$, $R = K_5$, and $g = 2$

We partition the edges of $R$ in two ways, both using two wavelengths:



# Example: $n = 5$, $R = K_5$, and $g = 2$

We partition the edges of $R$ in two ways, both using two wavelengths:



# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $O(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $O(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $\mathcal{O}(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $\mathcal{O}(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $\mathcal{O}(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $\mathcal{O}(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# State of the art (in the ring)

- Complexity:
  - TRAFFIC GROOMING (TG) is **NP-complete** considering $g$ as part of the input
    *(T.Chow et P.Lin, Networks'04)*
  - TG remains also NP-complete for fixed $g \geq 1$
    *(M.Shalom, W.Unger et S.Zaks, FUN'07)*
  - If $g$ is part of the input, TG **is not in APX**
    *(S.Huang, R.Dutta et G.N.Rouskas, IEEE JSAC'06)*
  - **Open problem:** inapproximability for fixed $g$
- Approximability:
  - Finding a $\sqrt{g}$-approximation is trivial (in polynomial time in $n$ and $g$)
  - The best approximation algorithm has ratio $\mathcal{O}(\log g)$, but the running time is exponential in $g$ (more precisely, $n^g$)
    *(M.Flammini et al., ISAAC'05)*
  - **Open problem:** *good* algorithm in polynomial time in $g$ and $n$ (i.e. when $g$ is also part of the input)

# Different models of TRAFFIC GROOMING (in COST 293):

- There are many different models of TRAFFIC GROOMING in the literature.
- We focus on a **particular model** (minimizing the number of ADMs)
- Other models: G/MPLS, light-trails, light-tours, multi-cast aggregation, ...
- Some of the people who is working in other models: T.Cinkler, M.Marciniak, J.-L.Marzo, F.Solano, J.Moulierac, A.Somani, ...

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Evolution of Traffic Grooming in COST 293:

- 2003 Coudert, Muñoz et al. *(ONDM'03)*:
  Statement of the problem in unidirectional rings (all-to-all case).
- 2004 Shalom and Zaks *(BROADNETS'04)*:
  $(\frac{10}{7} + \epsilon)$-approximation of grooming in rings for $g = 1$.
- 2005 Coudert, Muñoz et al. *(SIAM JDM'05)*:
  Grooming in unidirectional rings for $g = 6$ (all-to-all case).
- 2005 Coudert et al. *(SIROCCO'05)*:
  Grooming in unidirectional paths for $g = 1, 2$.
- 2005 Flammini, Moscadelli, Shalom and Zaks *(ISAAC'05)*:
  $\mathcal{O}(\log g)$-approximation of grooming in rings.
- 2006 Flammini, Moscadelli, Shalom, Zaks et al. *(WG'06)*:
  $\mathcal{O}(\log g)$-approximation of grooming in paths and stars.
- 2006 Coudert, Muñoz, Sau et al. *(ICTON'06)*:
  Grooming in bidirectional rings (all-to-all case).
- 2007 Shalom, Unger and Zaks *(FUN'07)*:
  Grooming in rings is NP-complete for fixed $g$.
- 2007 Amini, Pérennes and Sau *(ISAAC'07)*:
  Hardness and approximation of traffic grooming in rings and paths.

# Summing up: open questions

(1) Inapproximability (hardness) of TRAFFIC GROOMING for fixed $g$.

(2) *Good* algorithm for approximating TRAFFIC GROOMING, running in polynomial time in both $g$ and $n$, and with an approximation ratio not depending on $g$.

# Hardness of grooming for fixed $g$

- Now we are going to answer to the first open question:

**Theorem**

*Traffic grooming in the ring is* APX*-complete for fixed* $g \geq 1$.

**Theorem**

*Traffic grooming in the path is* APX*-complete for fixed* $g \geq 2$.

- To prove these results, we reduce traffic grooming to the following problem:

  **Finding the maximum number of edge-disjoint triangles in a graph.**

# Hardness of grooming for fixed $g$

- Now we are going to answer to the first open question:

**Theorem**

*Traffic grooming in the ring is* APX*-complete for fixed* $g \geq 1$.

**Theorem**

*Traffic grooming in the path is* APX*-complete for fixed* $g \geq 2$.

- To prove these results, we reduce traffic grooming to the following problem:

  **Finding the maximum number of edge-disjoint triangles in a graph.**

# Hardness of grooming for fixed $g$

- Now we are going to answer to the first open question:

**Theorem**

*Traffic grooming in the ring is* APX*-complete for fixed* $g \geq 1$.

**Theorem**

*Traffic grooming in the path is* APX*-complete for fixed* $g \geq 2$.

- To prove these results, we reduce traffic grooming to the following problem:

  **Finding the maximum number of edge-disjoint triangles in a graph.**

# MECT-B

- MAXIMUM BOUNDED EDGE COVERING BY TRIANGLES (MECT-B): Find the maximum number of edge-disjoint triangles in a graph of bounded degree $B$.
- The problem is NP-complete
*(I.Holyer, SIAM J.Comput'81)*
- Finding **node-disjoint** triangles is APX-complete
*(V.Kann, Inf. Proces. Let'91)*
- We have proved that MECT-B is also APX-complete

# MECT-B

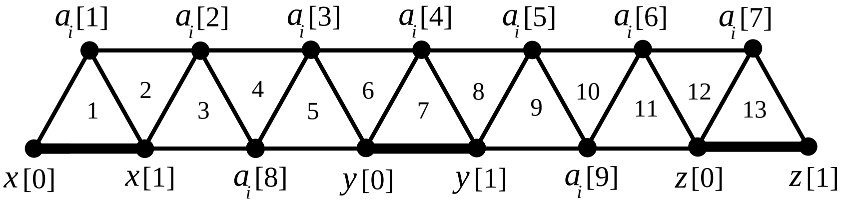
- MAXIMUM BOUNDED EDGE COVERING BY TRIANGLES (MECT-B): Find the maximum number of edge-disjoint triangles in a graph of bounded degree $B$.
- The problem is NP-complete *(I.Holyer, SIAM J.Comput'81)*
- Finding **node-disjoint** triangles is APX-complete *(V.Kann, Inf. Proces. Let'91)*
- We have proved that MECT-B is also APX-complete

# MECT-B

- MAXIMUM BOUNDED EDGE COVERING BY TRIANGLES (MECT-B): Find the maximum number of edge-disjoint triangles in a graph of bounded degree $B$.
- The problem is NP-complete
  *(I.Holyer, SIAM J.Comput'81)*
- Finding **node-disjoint** triangles is APX-complete
  *(V.Kann, Inf. Proces. Let'91)*
- We have proved that MECT-B is also APX-complete

# Idea of the proof

- Reduction from MAXIMUM BOUNDED COVERING BY 3-SETS (MAX 3SC-B).
- For each subset $c_i = \{x_i, y_i, z_i\}$ we build the following graph:

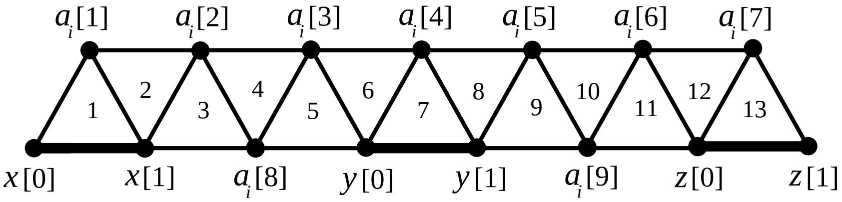



- We deduce that

$$OPT(\text{MECT-B}) \leq (18B + 1)OPT(\text{MAX 3SC-B})$$

# Idea of the proof

- Reduction from MAXIMUM BOUNDED COVERING BY 3-SETS (MAX 3SC-B).
- For each subset $c_i = \{x_i, y_i, z_i\}$ we build the following graph:

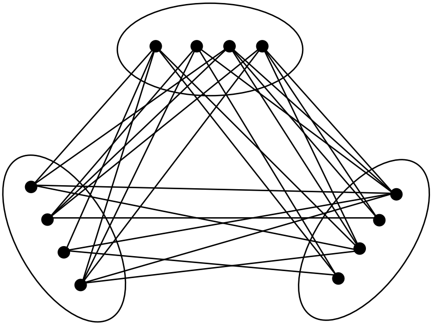



- We deduce that

$$OPT(\text{MECT-B}) \leq (18B+1)OPT(\text{MAX 3SC-B})$$

# APX-completeness of RING TRAFFIC GROOMING

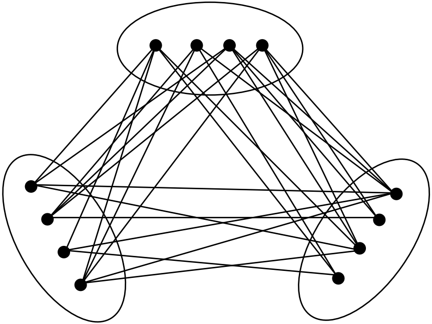
- Idea of the proof for $g = 1$: we take a tripartite request graph:



- The problem of traffic grooming is equivalent to finding the maximum number of edge-disjoint triangles
- Thus, traffic grooming is APX-complete

# APX-completeness of RING TRAFFIC GROOMING

- Idea of the proof for $g = 1$: we take a tripartite request graph:



- The problem of traffic grooming is equivalent to finding the maximum number of edge-disjoint triangles
- Thus, traffic grooming is APX-complete

# APX-completeness of PATH TRAFFIC GROOMING

- The problem is in $P$ for $g = 1$
  *(J.-C.Bermond, L.Braud and D.Coudert, SIROCCO'05)*

- The complexity for fixed $g \geq 2$ (even if P/NP?) has been an open problem for a while

- We have proved that the problem is APX-complete for fixed $g \geq 2$

# APX-completeness of PATH TRAFFIC GROOMING

- The problem is in $P$ for $g = 1$
  *(J.-C.Bermond, L.Braud and D.Coudert, SIROCCO'05)*

- The complexity for fixed $g \geq 2$ (even if P/NP?) has been an open problem for a while

- We have proved that the problem is APX-complete for fixed $g \geq 2$

# Approximation algorithm for TRAFFIC GROOMING

- Now we are going to answer to the second open question:

  **approximation algorithm** (for rings and paths) with running time polynomial in both $n$ and $g$, with an approximation ratio depending only on $n$:

**Theorem**

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

# Approximation algorithm for TRAFFIC GROOMING

- Now we are going to answer to the second open question:

  **approximation algorithm** (for rings and paths) with running time polynomial in both $n$ and $g$, with an approximation ratio depending only on $n$:

**Theorem**

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

# Idea of the algorithm: first step

- Divide the request set into $\log n$ classes $C_i$, $i = 0, \ldots, \log n - 1$ s.t. in each class $C_i$ the length of the requests is in $[2^i, 2^{i+1})$:

$$
\begin{aligned}
C_0 &\rightarrow \text{length in the interval } [1,2) \\
C_1 &\rightarrow \text{length in the interval } [2,4) \\
C_2 &\rightarrow \text{length in the interval } [4,8) \\
&\ldots \\
C_{\log n-1} &\rightarrow \text{length in the interval } [\frac{n}{2}, n)
\end{aligned}
$$

# First step of the algorithm (II)

- For each class $C_i$ $\longrightarrow$ the ring can be divided into intervals of length $2^i$ s.t. the **only requests are between consecutive intervals.**
- we obtain $\frac{n}{2^i}$ subproblems for each class:
each one consists in finding an optimal solution in a bipartite graph of size $2 \cdot 2^i$.

  BIPARTITE TRAFFIC GROOMING

  **Input:** A bipartite graph $R$, and a grooming factor $g$.
  **Output:** Partition of the edges of $R$ into subgraphs $R_\omega$ with at most $g$ edges, $1 \leq \omega \leq W$.
  **Objective:** Minimize $\sum_{\omega=1}^{W} |V(R_\omega)|$.

- Solve all these BIPARTITE TRAFFIC GROOMING subproblems independently, and output the union of all solutions.

# First step of the algorithm (II)

- For each class $C_i \quad \rightarrow \quad$ the ring can be divided into intervals of length $2^i$ s.t. the **only requests are between consecutive intervals.**

- we obtain $\frac{n}{2^i}$ subproblems for each class:
  each one consists in finding an optimal solution in a bipartite graph of size $2 \cdot 2^i$.

  BIPARTITE TRAFFIC GROOMING

  **Input:** A bipartite graph $R$, and a grooming factor $g$.
  **Output:** Partition of the edges of $R$ into subgraphs $R_\omega$ with at most $g$ edges, $1 \leq \omega \leq W$.
  **Objective:** Minimize $\sum_{\omega=1}^{W} |V(R_\omega)|$.

- Solve all these BIPARTITE TRAFFIC GROOMING subproblems independently, and output the union of all solutions.

# First step of the algorithm (II)

- For each class $C_i \quad \rightarrow \quad$ the ring can be divided into intervals of length $2^i$ s.t. the **only requests are between consecutive intervals.**

- we obtain $\frac{n}{2^i}$ subproblems for each class:
  each one consists in finding an optimal solution in a bipartite graph of size $2 \cdot 2^i$.

  BIPARTITE TRAFFIC GROOMING

  **Input:** A bipartite graph $R$, and a grooming factor $g$.
  **Output:** Partition of the edges of $R$ into subgraphs $R_\omega$ with at most $g$ edges, $1 \leq \omega \leq W$.
  **Objective:** Minimize $\sum_{\omega=1}^{W} |V(R_\omega)|$.

- Solve all these BIPARTITE TRAFFIC GROOMING subproblems independently, and output the union of all solutions.

# First step of the algorithm (II)

- For each class $C_i$ $\rightarrow$ the ring can be divided into intervals of length $2^i$ s.t. the **only requests are between consecutive intervals**.

- we obtain $\frac{n}{2^i}$ subproblems for each class:
  each one consists in finding an optimal solution in a bipartite graph of size $2 \cdot 2^i$.

  BIPARTITE TRAFFIC GROOMING

  **Input:** A bipartite graph $R$, and a grooming factor $g$.
  **Output:** Partition of the edges of $R$ into subgraphs $R_\omega$ with at most $g$ edges, $1 \leq \omega \leq W$.
  **Objective:** Minimize $\sum_{\omega=1}^{W} |V(R_\omega)|$.

- Solve all these BIPARTITE TRAFFIC GROOMING subproblems independently, and output the union of all solutions.

# Second step of the algorithm

- The ***density*** $\rho(G)$ of a graph $G = (V, E)$: is its edges-to-vertices ratio, that is:

$$\rho(G) := \frac{|E(G)|}{|V(G)|}$$

- More generally, for any subset $S \subset V$, we call *density* of $S$, $\rho_G(S)$ or simply $\rho(S)$, to the density of the induced graph on $S$, i.e. $\rho(S) := \rho(G[S])$.

- We use theDENSE $k$-SUBGRAPH optimization problem:

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

*(U.Feige, D.Peleg and G.Kortsarz, Algorithmica'01)*

# Second step of the algorithm

- The ***density*** $\rho(G)$ of a graph $G = (V, E)$: is its edges-to-vertices ratio, that is:
$$\rho(G) := \frac{|E(G)|}{|V(G)|}$$
- More generally, for any subset $S \subset V$, we call *density* of $S$, $\rho_G(S)$ or simply $\rho(S)$, to the density of the induced graph on $S$, i.e. $\rho(S) := \rho(G[S])$.
- We use theDENSE $k$-SUBGRAPH optimization problem:

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

*(U.Feige, D.Peleg and G.Kortsarz, Algorithmica'01)*

# Second step of the algorithm (II)

- To solve each BIPARTITE TRAFFIC GROOMING subproblem in a bipartite graph $R$:

proceed **greedily** (until all edges are covered),
by finding at step $i$ a **subgraph**

$$R_i \subseteq G \setminus (R_1 \cup \cdots \cup R_{i-1})$$

**with at most** $g$ **edges** in the following way:

For each $k = 2, \ldots, 2g$ find a subgraph

$$B_k \subseteq R \setminus (R_1 \cup \cdots \cup R_{i-1})$$

using the best algorithm for the **DENSE** $k$**-SUBGRAPH** problem.
*(U.Feige, D.Peleg and G.Kortsarz, Algorithmica'01)*

- Now we have to choose one of these $B_k$.

# Second step of the algorithm (II)

- To solve each BIPARTITE TRAFFIC GROOMING subproblem in a bipartite graph $R$:

  proceed **greedily** (until all edges are covered),
  by finding at step $i$ a **subgraph**

  $$R_i \subseteq G \setminus (R_1 \cup \cdots \cup R_{i-1})$$

  **with at most $g$ edges** in the following way:

  For each $k = 2, \ldots, 2g$ find a subgraph

  $$B_k \subseteq R \setminus (R_1 \cup \cdots \cup R_{i-1})$$

  using the best algorithm for the **DENSE $k$-SUBGRAPH** problem.
  *(U.Feige, D.Peleg and G.Kortsarz, Algorithmica'01)*

- Now we have to choose one of these $B_k$.

# Second step of the algorithm (II)

- To solve each BIPARTITE TRAFFIC GROOMING subproblem in a bipartite graph $R$:

  proceed **greedily** (until all edges are covered),
  by finding at step $i$ a **subgraph**

  $$R_i \subseteq G \setminus (R_1 \cup \cdots \cup R_{i-1})$$

  **with at most $g$ edges** in the following way:

  For each $k = 2, \ldots, 2g$ find a subgraph

  $$B_k \subseteq R \setminus (R_1 \cup \cdots \cup R_{i-1})$$

  using the best algorithm for the **DENSE $k$-SUBGRAPH** problem.
  *(U.Feige, D.Peleg and G.Kortsarz, Algorithmica'01)*

- Now we have to choose one of these $B_k$.

# Second step of the algorithm (III)

- If for some $k^*$,

$$|E(B_{k^*})| > g \text{ , and } |E(B_i)| \leq g \text{ for all } i < k^*,$$

**remove** $|E(B_k^*)| - g$ **arbitrary edges** of $B_{k^*}$, and replace $B_k^*$ with this new graph.

Stop the search at $k^*$, and **output the densest graph** among $B_2, \ldots, B_{k^*-1}, B_{k^*}$.

- If not, output the densest subgraph among $B_2, \ldots, B_{2g}$.

# Second step of the algorithm (III)

- If for some $k^*$,

$$|E(B_{k^*})| > g \ \text{, and} \ |E(B_i)| \leq g \ \text{ for all } \ i < k^*,$$

**remove** $|E(B_k^*)| - g$ **arbitrary edges** of $B_{k^*}$, and replace $B_k^*$ with this new graph.

Stop the search at $k^*$, and **output the densest graph** among $B_2, \ldots, B_{k^*-1}, B_{k^*}$.

- If not, output the densest subgraph among $B_2, \ldots, B_{2g}$.

# Second step of the algorithm (III)

- If for some $k^*$,

$$|E(B_{k^*})| > g \ \text{, and} \ |E(B_i)| \leq g \ \text{ for all } \ i < k^*,$$

**remove** $|E(B_k^*)| - g$ **arbitrary edges** of $B_{k^*}$, and replace $B_k^*$ with this new graph.

Stop the search at $k^*$, and **output the densest graph** among $B_2, \ldots, B_{k^*-1}, B_{k^*}$.

- If not, output the densest subgraph among $B_2, \ldots, B_{2g}$.

# Approximation ratio?

- *Recall*: $n$ is the number of nodes, and $g$ is the grooming factor.
- **Step 1**: divide the request set into subsets (according to the length), and solve each problem independently.

  We loose a factor $2 \log n$
- **Step 2**: proceed greedily by removing a subgraph found using the algorithm for DENSE $k$-SUBGRAPH (removing edges if necessary).

  We loose a factor $2 \log n$ due to the *greedy* approach, and a factor $2n^{1/3}$ due to DENSE $k$-SUBGRAPH
- So, the approximation ratio of the algorithm is at most:

$$2 \log n \cdot 2 \log n \cdot 2n^{1/3} = 8n^{1/3} \log^2 n$$

- **Remark:** if the request graph is $H$-minor free, then the approximation ratio becomes $\mathcal{O}(\log^2 n)$.

# Approximation ratio?

- *Recall*: $n$ is the number of nodes, and $g$ is the grooming factor.
- **Step 1**: divide the request set into subsets (according to the length), and solve each problem independently.

  We loose a factor $2 \log n$

- **Step 2**: proceed greedily by removing a subgraph found using the algorithm for DENSE $k$-SUBGRAPH (removing edges if necessary).

  We loose a factor $2 \log n$ due to the *greedy* approach, and a factor $2n^{1/3}$ due to DENSE $k$-SUBGRAPH

- So, the approximation ratio of the algorithm is at most:

$$2 \log n \cdot 2 \log n \cdot 2n^{1/3} = 8n^{1/3} \log^2 n$$

- **Remark:** if the request graph is $H$-minor free, then the approximation ratio becomes $\mathcal{O}(\log^2 n)$.

# Approximation ratio?

- *Recall*: $n$ is the number of nodes, and $g$ is the grooming factor.
- **Step 1**: divide the request set into subsets (according to the length), and solve each problem independently.

  We loose a factor $2 \log n$
- **Step 2**: proceed greedily by removing a subgraph found using the algorithm for DENSE $k$-SUBGRAPH (removing edges if necessary).

  We loose a factor $2 \log n$ due to the *greedy* approach, and a factor $2n^{1/3}$ due to DENSE $k$-SUBGRAPH
- So, the approximation ratio of the algorithm is at most:

$$2 \log n \cdot 2 \log n \cdot 2n^{1/3} = 8n^{1/3} \log^2 n$$

- **Remark:** if the request graph is $H$-minor free, then the approximation ratio becomes $\mathcal{O}(\log^2 n)$.

# Approximation ratio?

- *Recall*: $n$ is the number of nodes, and $g$ is the grooming factor.
- **Step 1**: divide the request set into subsets (according to the length), and solve each problem independently.

  We loose a factor $2 \log n$

- **Step 2**: proceed greedily by removing a subgraph found using the algorithm for DENSE $k$-SUBGRAPH (removing edges if necessary).

  We loose a factor $2 \log n$ due to the *greedy* approach, and a factor $2n^{1/3}$ due to DENSE $k$-SUBGRAPH

- So, the approximation ratio of the algorithm is at most:

$$2 \log n \cdot 2 \log n \cdot 2n^{1/3} = 8n^{1/3} \log^2 n$$

- **Remark:** if the request graph is $H$-minor free, then the approximation ratio becomes $\mathcal{O}(\log^2 n)$.

# Approximation ratio?

- *Recall*: $n$ is the number of nodes, and $g$ is the grooming factor.
- **Step 1**: divide the request set into subsets (according to the length), and solve each problem independently.

  We loose a factor $2 \log n$
- **Step 2**: proceed greedily by removing a subgraph found using the algorithm for DENSE $k$-SUBGRAPH (removing edges if necessary).

  We loose a factor $2 \log n$ due to the *greedy* approach, and a factor $2n^{1/3}$ due to DENSE $k$-SUBGRAPH
- So, the approximation ratio of the algorithm is at most:

$$2 \log n \cdot 2 \log n \cdot 2n^{1/3} = 8n^{1/3} \log^2 n$$

- **Remark:** if the request graph is $H$-minor free, then the approximation ratio becomes $\mathcal{O}(\log^2 n)$.

# Conclusions and further research

- We have proved that TRAFFIC GROOMING for a fixed grooming factor $g$ does not accept to be approximated within any constant factor.

- We have exhibited a polynomial-time approximation algorithm for TRAFFIC GROOMING when both the number of nodes and the grooming factor belong to the input.

- Further research (for both PATH and RING):
  - **For fixed** $g$: improve the best constant-factor approximation algorithm.
    So far is $\mathcal{O}(\log g)$
  - **For** $g$ **belonging to the input**: improve the algorithm and/or the best existing hardness result.
    So far is not APX, we conjecture that it is $n^{\varepsilon}$ for some constant $\varepsilon > 0$

# Conclusions and further research

- We have proved that TRAFFIC GROOMING for a fixed grooming factor $g$ does not accept to be approximated within any constant factor.
- We have exhibited a polynomial-time approximation algorithm for TRAFFIC GROOMING when both the number of nodes and the grooming factor belong to the input.
- Further research (for both PATH and RING):
  - **For fixed** $g$: improve the best constant-factor approximation algorithm.
    So far is $\mathcal{O}(\log g)$
  - **For** $g$ **belonging to the input**: improve the algorithm and/or the best existing hardness result.
    So far is not APX, we conjecture that it is $n^{\varepsilon}$ for some constant $\varepsilon > 0$

# Conclusions and further research

- We have proved that TRAFFIC GROOMING for a fixed grooming factor $g$ does not accept to be approximated within any constant factor.
- We have exhibited a polynomial-time approximation algorithm for TRAFFIC GROOMING when both the number of nodes and the grooming factor belong to the input.
- Further research (for both PATH and RING):
  - **For fixed** $g$: improve the best constant-factor approximation algorithm.
    So far is $\mathcal{O}(\log g)$
  - **For** $g$ **belonging to the input**: improve the algorithm and/or the best existing hardness result.
    So far is not APX, we conjecture that it is $n^{\varepsilon}$ for some constant $\varepsilon > 0$

# Conclusions and further research

- We have proved that TRAFFIC GROOMING for a fixed grooming factor $g$ does not accept to be approximated within any constant factor.
- We have exhibited a polynomial-time approximation algorithm for TRAFFIC GROOMING when both the number of nodes and the grooming factor belong to the input.
- Further research (for both PATH and RING):
  - **For fixed** $g$: improve the best constant-factor approximation algorithm.
    So far is $\mathcal{O}(\log g)$
  - **For** $g$ **belonging to the input**: improve the algorithm and/or the best existing hardness result.
    So far is not APX, we conjecture that it is $n^{\varepsilon}$ for some constant $\varepsilon > 0$

# Conclusions and further research

- We have proved that TRAFFIC GROOMING for a fixed grooming factor $g$ does not accept to be approximated within any constant factor.
- We have exhibited a polynomial-time approximation algorithm for TRAFFIC GROOMING when both the number of nodes and the grooming factor belong to the input.
- Further research (for both PATH and RING):
  - **For fixed** $g$: improve the best constant-factor approximation algorithm.
    So far is $\mathcal{O}(\log g)$
  - **For** $g$ **belonging to the input**: improve the algorithm and/or the best existing hardness result.
    So far is not APX, we conjecture that it is $n^{\varepsilon}$ for some constant $\varepsilon > 0$

Thanks!