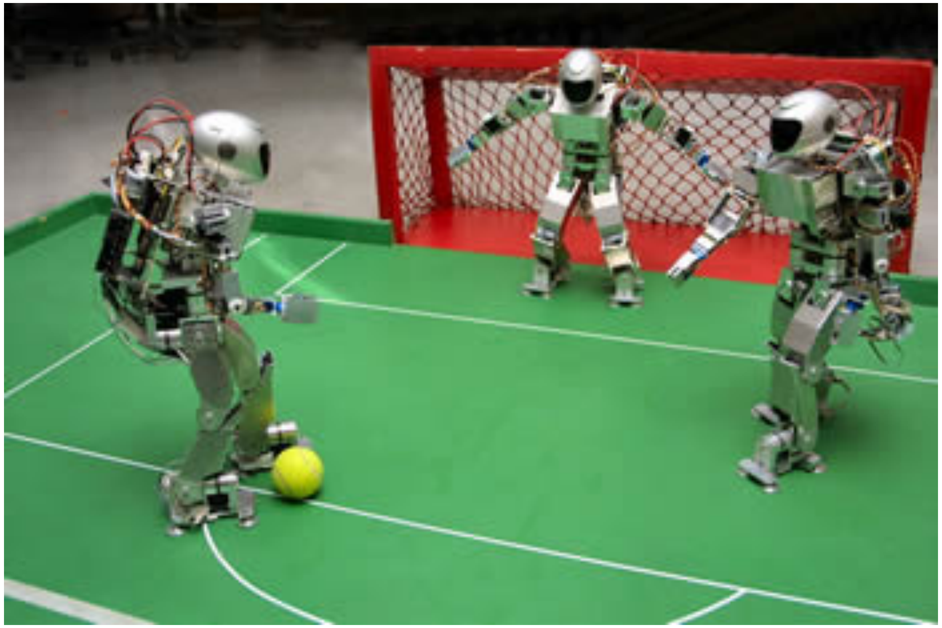
Graph Routing Problems: Approximation, Hardness, and Graph-Theoretic Insights

Julia Chuzhoy

Toyota Technological Institute at Chicago

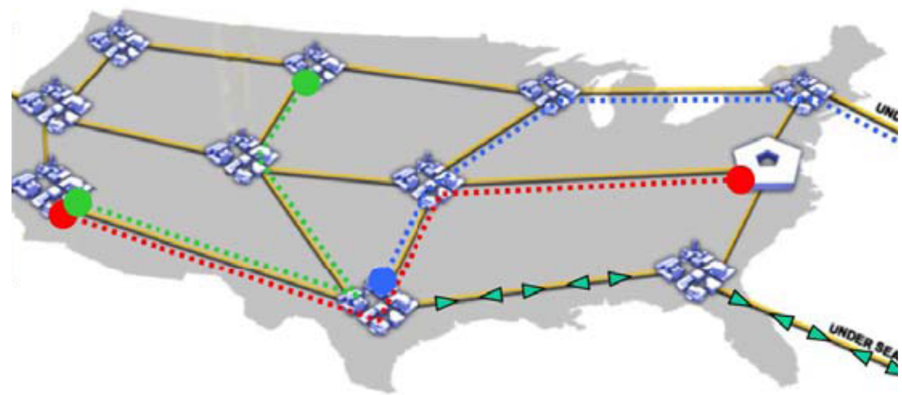

ROAD
CLOSED
DETOUR


VLSI
9136AV 195432
VL82C106-FC
3066F4000
COPR.1991




ROAD
CLOSED
DETOUR


VLSI
9136AV 195432
VL82C106-FC
3066F4000
COPR.1991



Graph Routing Problems

maximum s-t flow

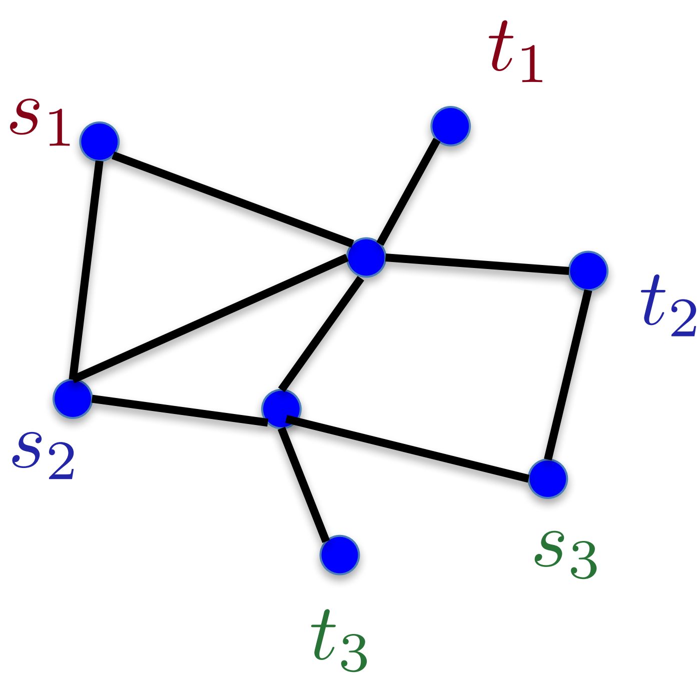
maximum multicommodity flow

maximum node-disjoint paths (NDP)

Node-Disjoint Paths (NDP)

Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

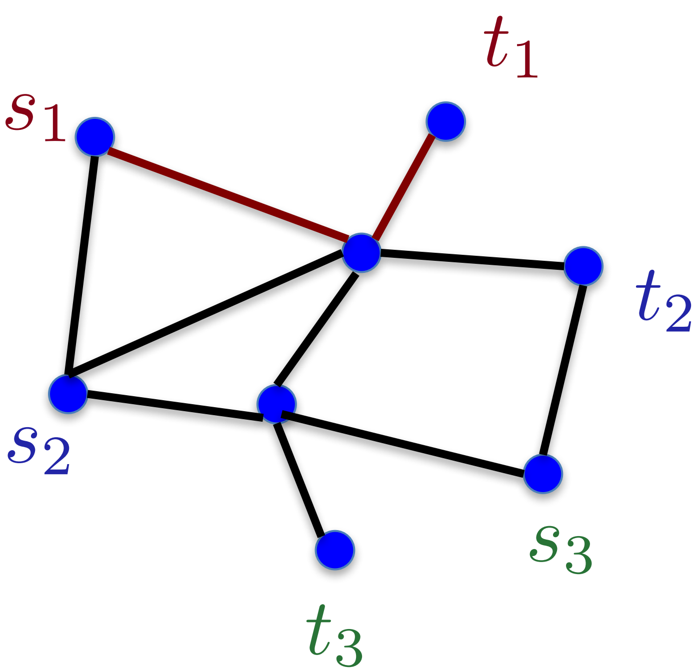
Goal: Route as many pairs as possible via node-disjoint paths



Node-Disjoint Paths (NDP)

Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

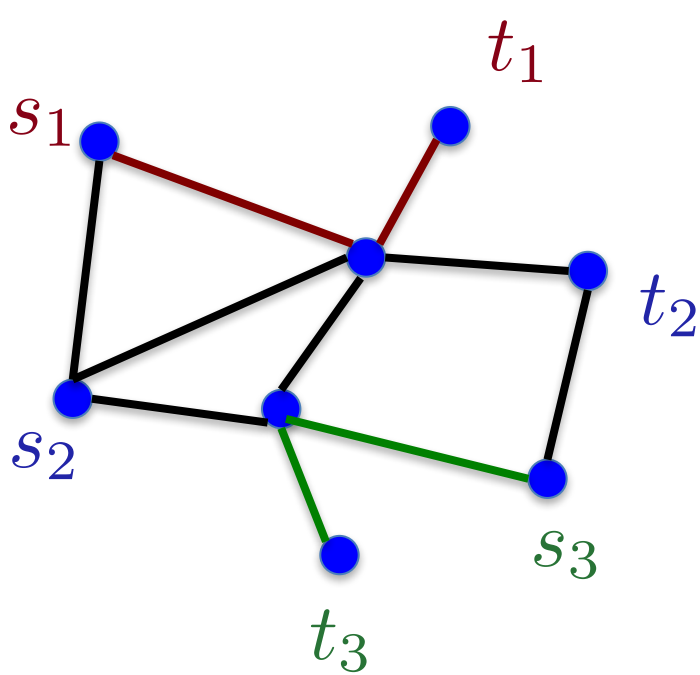
Goal: Route as many pairs as possible via node-disjoint paths



Node-Disjoint Paths (NDP)

Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

Goal: Route as many pairs as possible via node-disjoint paths



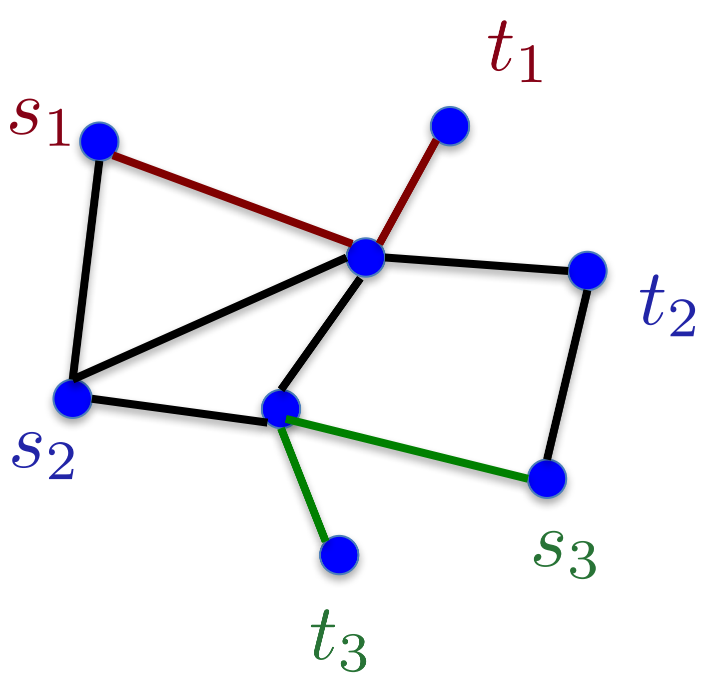
Solution value: 2

OPT: value of best possible solution

Node-Disjoint Paths (NDP)

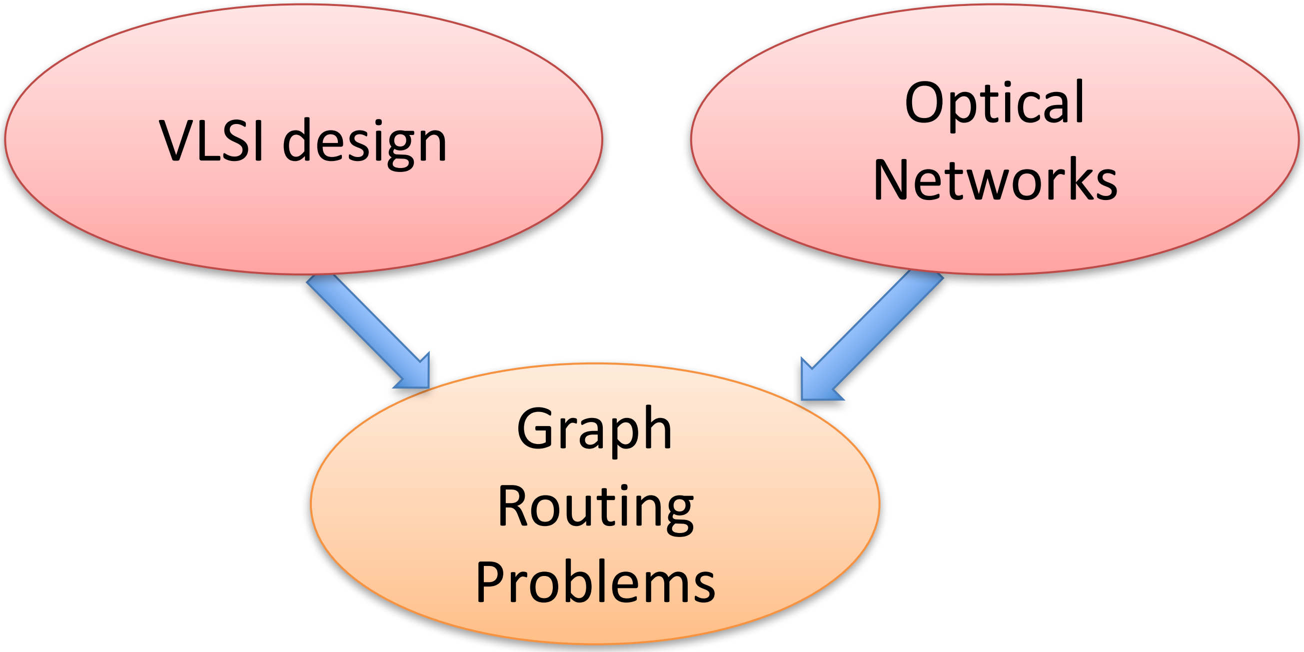
Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

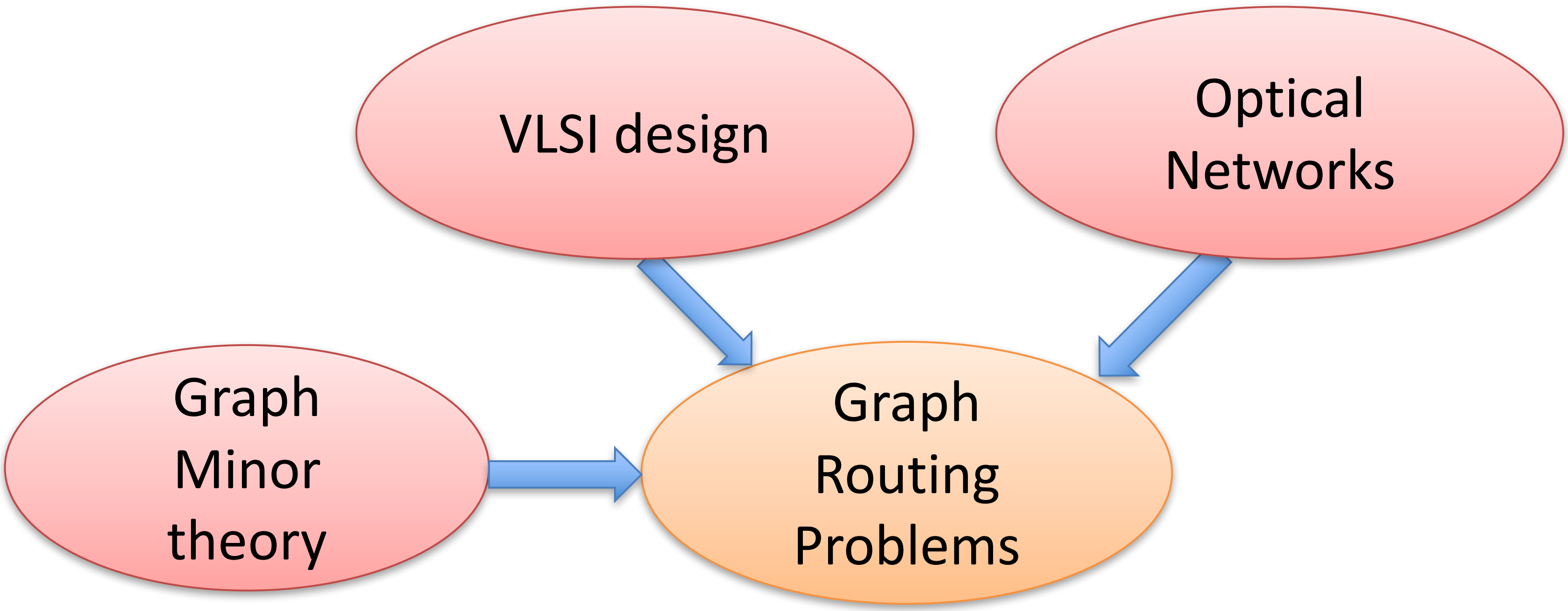
Goal: Route as many pairs as possible via node-disjoint paths

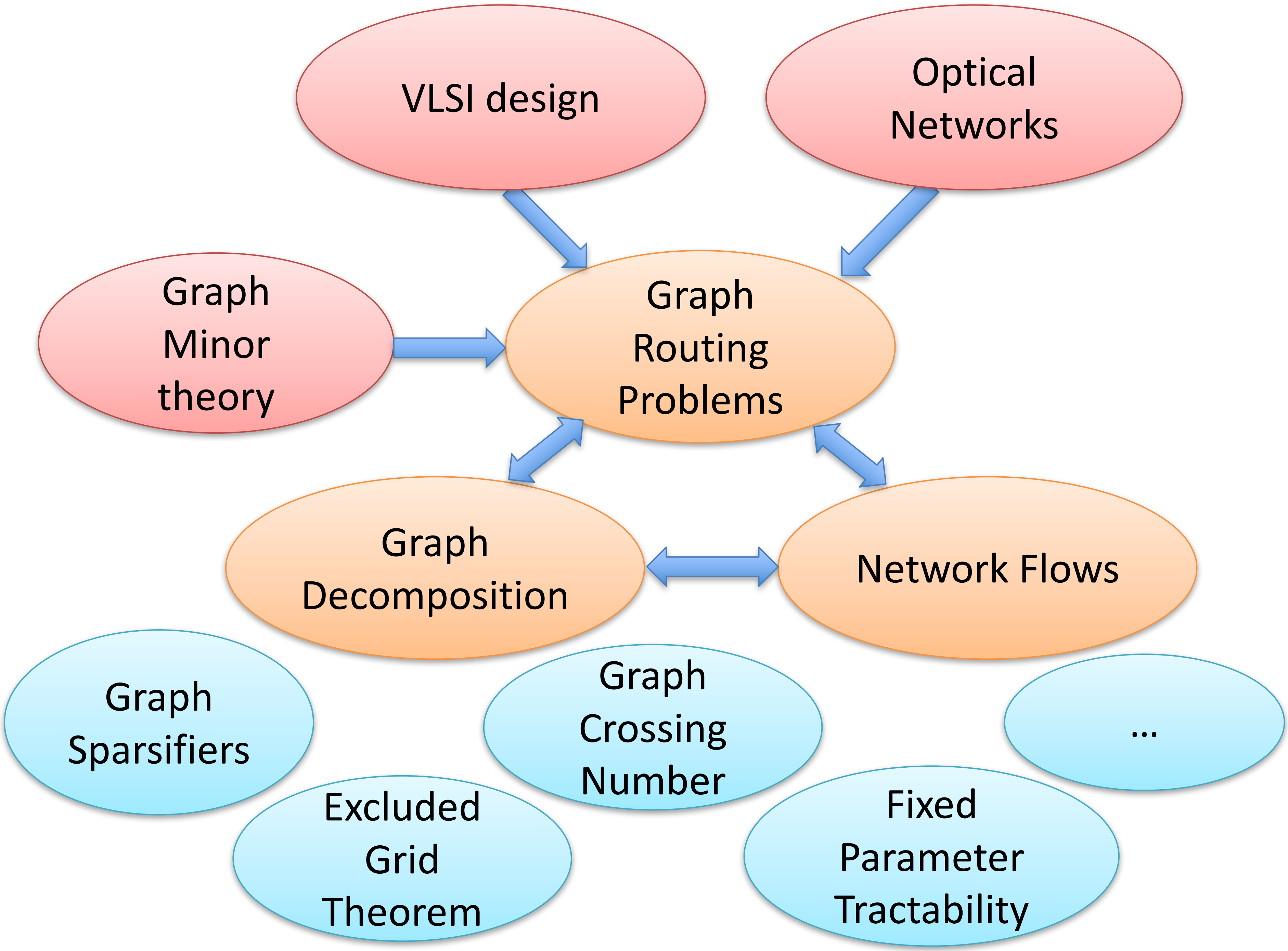


Solution value: 2

Edge-disjoint Paths (EDP): paths must be edge-disjoint


VLSI design
Optical Networks
Graph Routing Problems


VLSI design
Optical Networks
Graph Minor theory
Graph Routing Problems

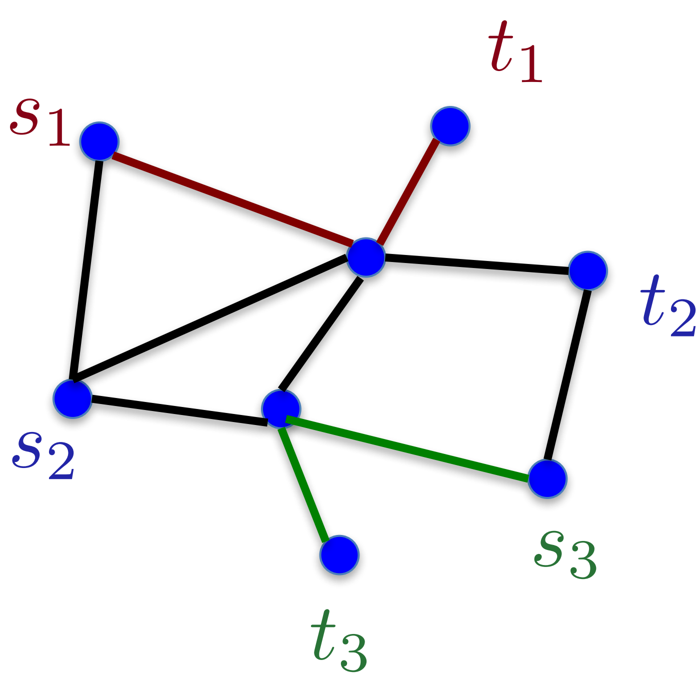

VLSI design
Optical Networks
Graph Minor theory
Graph Routing Problems
Graph Decomposition
Network Flows
Graph Sparsifiers
Graph Crossing Number
...
Excluded Grid Theorem
Fixed Parameter Tractability

Node-Disjoint Paths (NDP)

Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

Goal: Route as many pairs as possible via node-disjoint paths

terminals

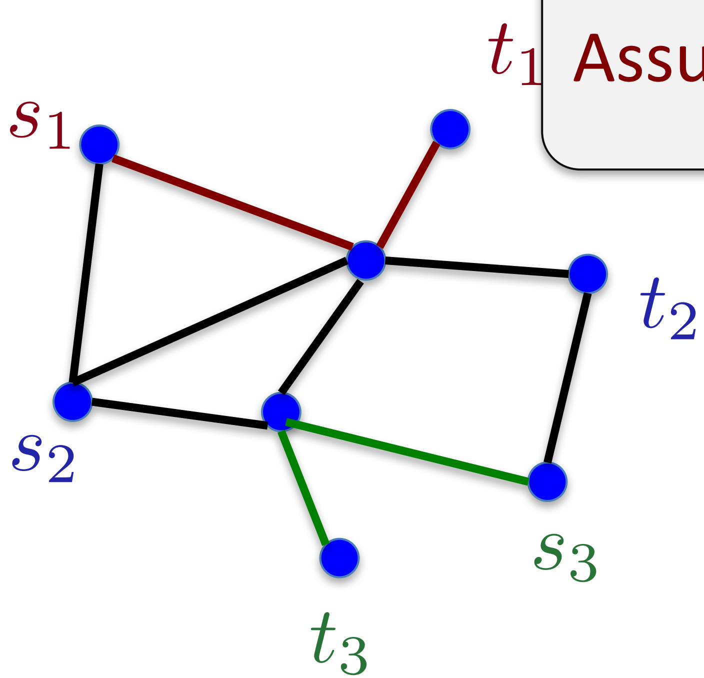


Node-Disjoint Paths (NDP)

Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

Goal: Route as many pairs as possible via node-disjoint paths

Assumption: All terminals are distinct



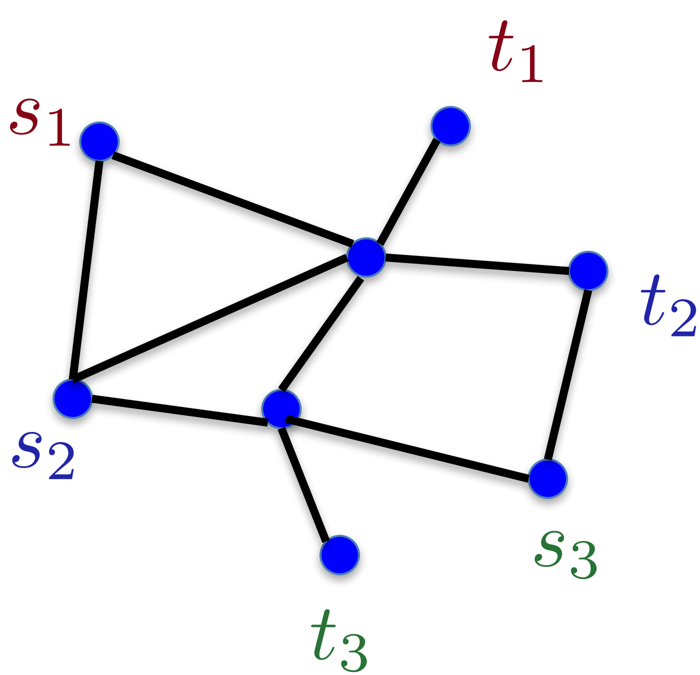
n – number of graph vertices

k – number of demand pairs

Node-Disjoint Paths (NDP)

Input: Graph G, source-sink pairs $(s_1,t_1),\ldots,(s_k,t_k)$.

Goal: Route as many pairs as possible via node-disjoint paths



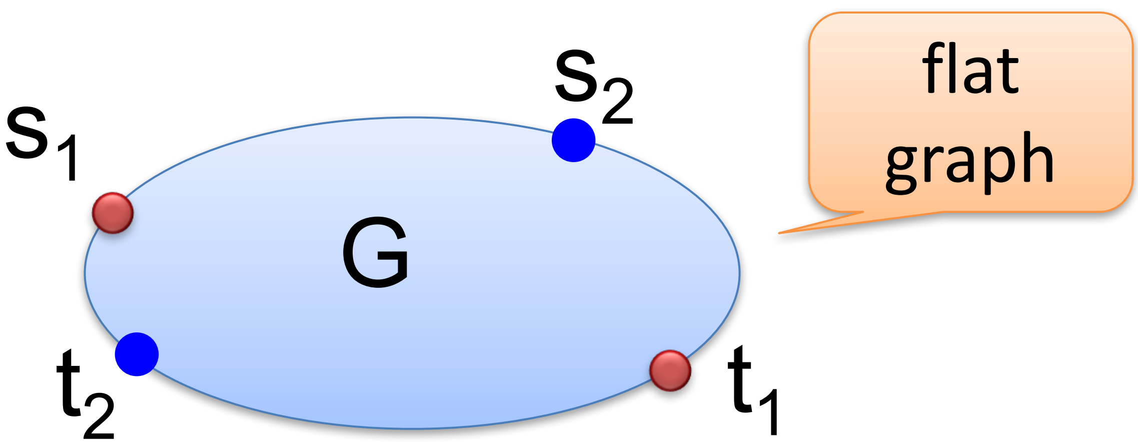
Can we solve it efficiently?

k=1? ✔

k=2?

NDP with k=2

- NP-hard in directed graphs [Fortune, Hopcroft, Wyllie '80]
- Efficiently solvable in undirected graphs [Jung ‘70, Shiloach ’80, Thomassen ‘80, Robertson-Seymour ’90]



Larger k?

Larger k?

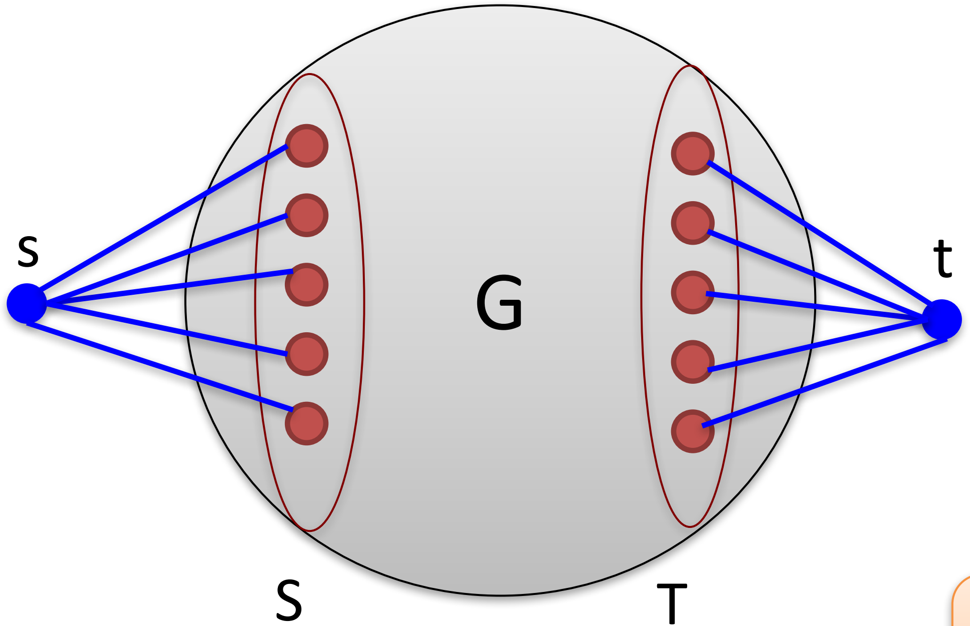
- Constant k: efficiently solvable [Robertson, Seymour '90]
  - Running time: f(k)•n^2 [Kawarabayashi, Kobayashi, Reed '12]

$$f(k) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{k}}}}}}$$

Larger k?

- Constant k: efficiently solvable [Robertson, Seymour ’90]
  - Running time: $f(k) \cdot n^2$ [Kawarabayashi, Kobayashi, Reed ‘12]
- NP-hard when k is part of input [Knuth, Karp ’74]

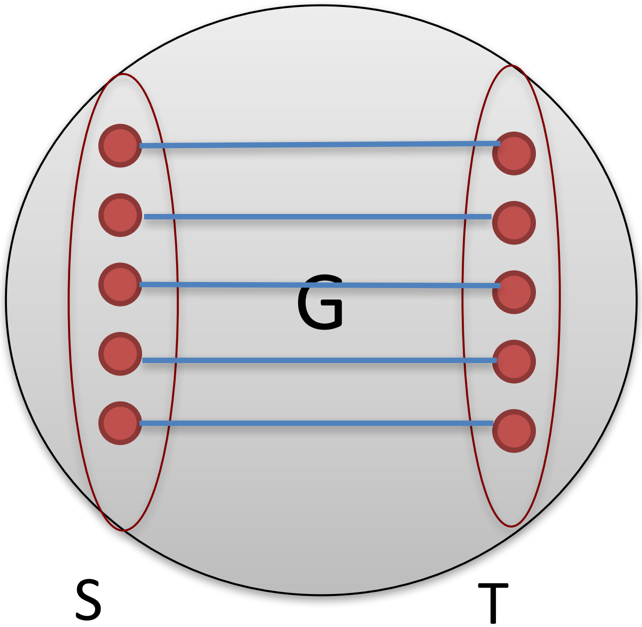
Example



Max s-t flow

- Set of demand pairs is SxT: can solve efficiently

Example



- Set of demand pairs is SxT: can solve efficiently
- Demand pairs are a specific matching between S and T: NP-hard

Dealing with NP-Hardness

An α-approximation algorithm:

- efficient algorithm
- always produces solutions routing at least OPT/α demand pairs.

Dealing with NP-Hardness

An α-approximation algorithm:

- efficient algorithm
- always produces solutions routing at least OPT/α demand pairs.

Dealing with NP-Hardness

An α-approximation algorithm:

- efficient algorithm
- always produces solutions routing at least OPT/α demand pairs.

optimum solution value

On Approximation Factors

- A simple way to compare algorithms
  - $\alpha=1+\varepsilon$
  - $\alpha=2$
  - $\alpha=O(\log n)$
  - $\alpha=O(\sqrt{n})$
  - …

On Approximation Factors

- A simple way to compare algorithms
- Design algorithms with good approximation factors α
- Establish best possible approximation factor α for a given problem

Hardness of approximation results

On Approximation Factors

- A simple way to compare algorithms
- Design algorithms with good approximation factors α
- Establish best possible approximation factor α for a given problem

Hardness of approximation results

On Approximation Factors

- A simple way to compare algorithms
- Design algorithms with good approximation factors α
- Establish best possible a
  for a given problem

Goal: powerful, simple algorithmic techniques with provable bounds

On Approximation Factors

- A simple way to compare algorithms
- Design algorithms with good approximation factors α
- Establish best possible a tor α for a given problem

Better models for real-life problems

Understanding what makes a problem difficult

On Approximation Factors

- A simple way to compare algorithms
- Design algorithms with good approximation factors α
- Establish best possible approximation factor α for a given problem

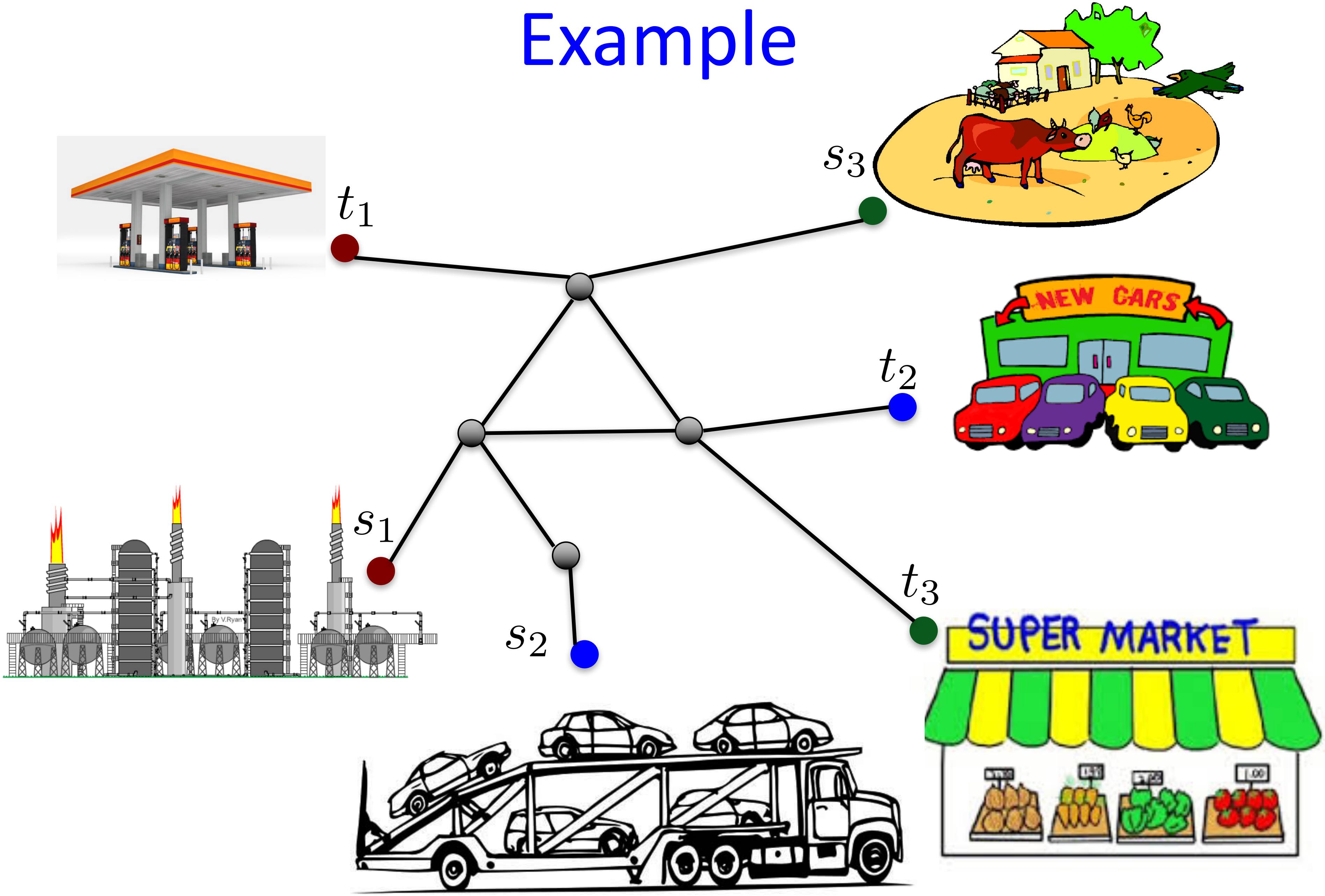
Dealing with NP-Hardness

An α-approximation algorithm:

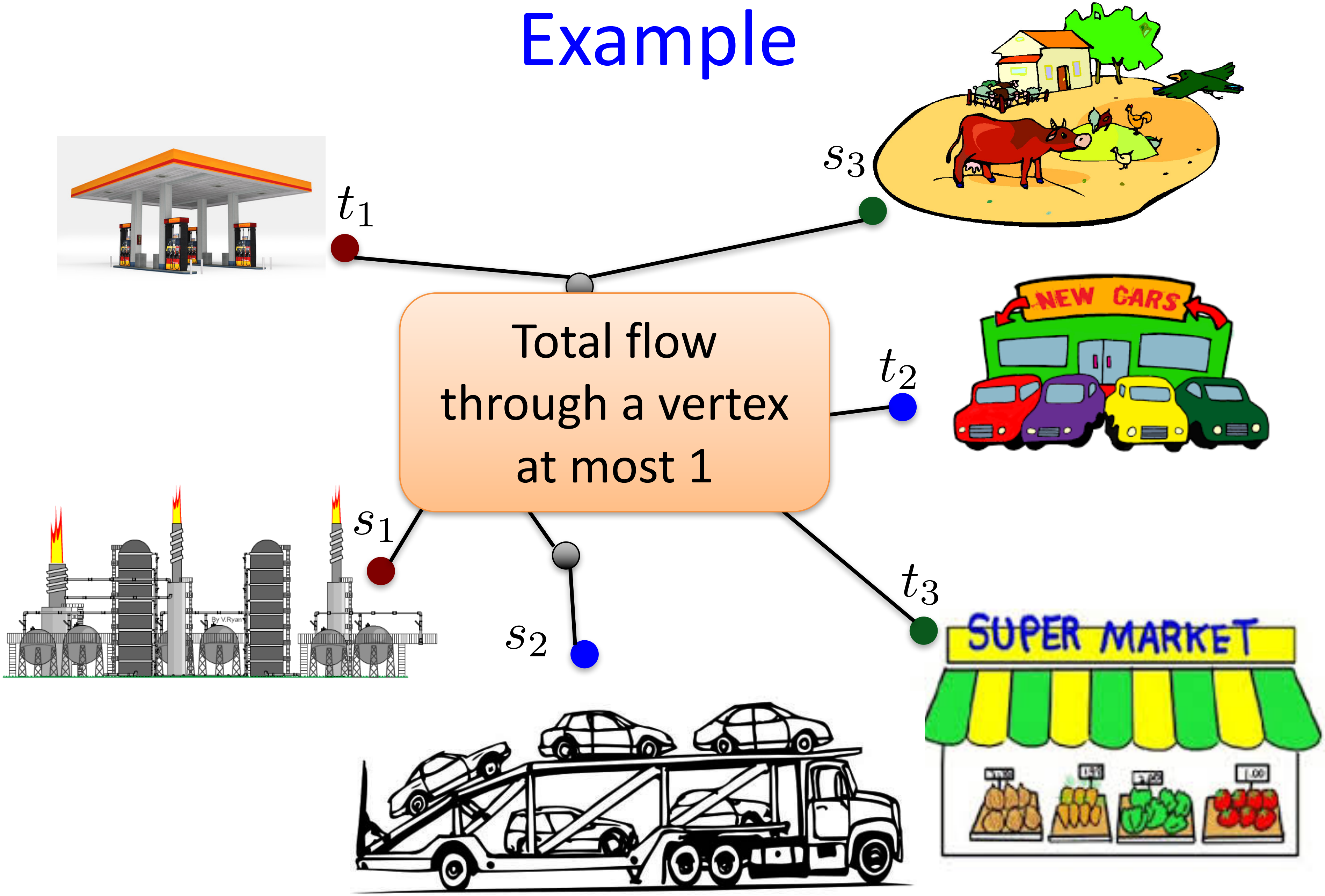
- efficient algorithm
- always produces solutions routing at least OPT/α demand pairs.

Multicommodity Flow relaxation: send as much flow as possible between the s_i-t_i pairs.

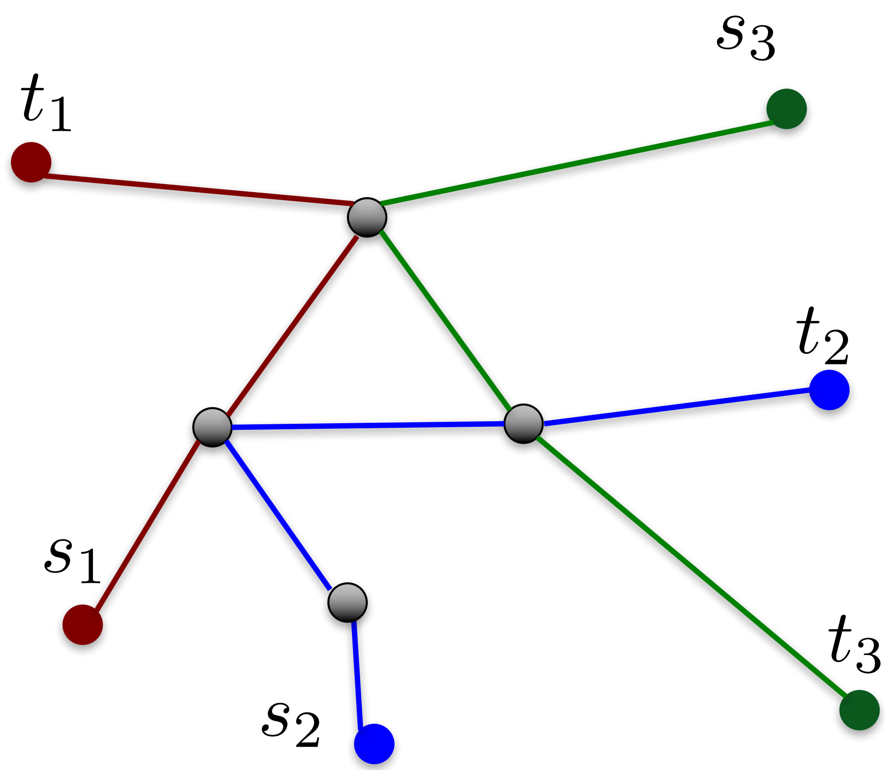
Example



Example


s_3
t_1
NEW CARS
Total flow through a vertex at most 1
t_2
s_1
s_2
t_3
SUPER MARKET

Example



NDP solution value: 1

- send ½ flow unit on each of the 3 paths
- solution value: 3/2

Multicommodity Flows

- Can be computed efficiently
- $OPT_{flow} \geq OPT$

multicommodity flow LP-relaxation

fractional solution

integral solution

Multicommodity Flows

- Can be computed efficiently

multicommodity flow LP-relaxation

- $OPT_{flow} \geq OPT$

- Use the flow to find integral routing of at least OPT_{flow}/α demand pairs

α-approximation algorithm

LP-rounding technique

Approximation Algorithm [Kolliopoulos, Stein '98]

While there is a path P with f(P)>0:

- Add such shortest path P to the solution
- For each path P' sharing vertices with P, set f(P') to 0

Approximation Algorithm [Kolliopoulos, Stein '98]

While there is a path P with f(P)>0:

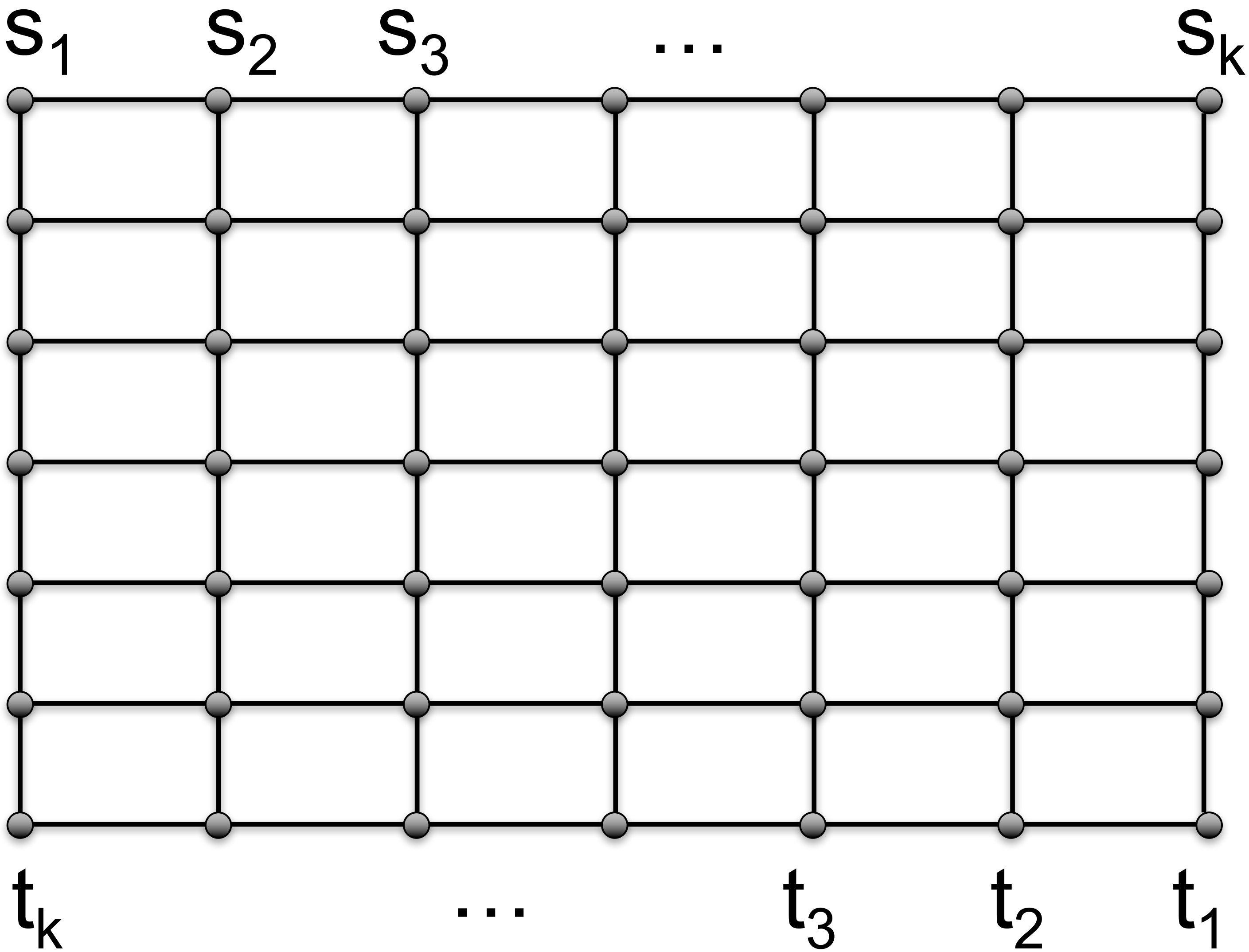
- Add such shortest path P to the solution
- For each path P' sharing vertices with P, set f(P') to 0

$O(\sqrt{n})$-approximation

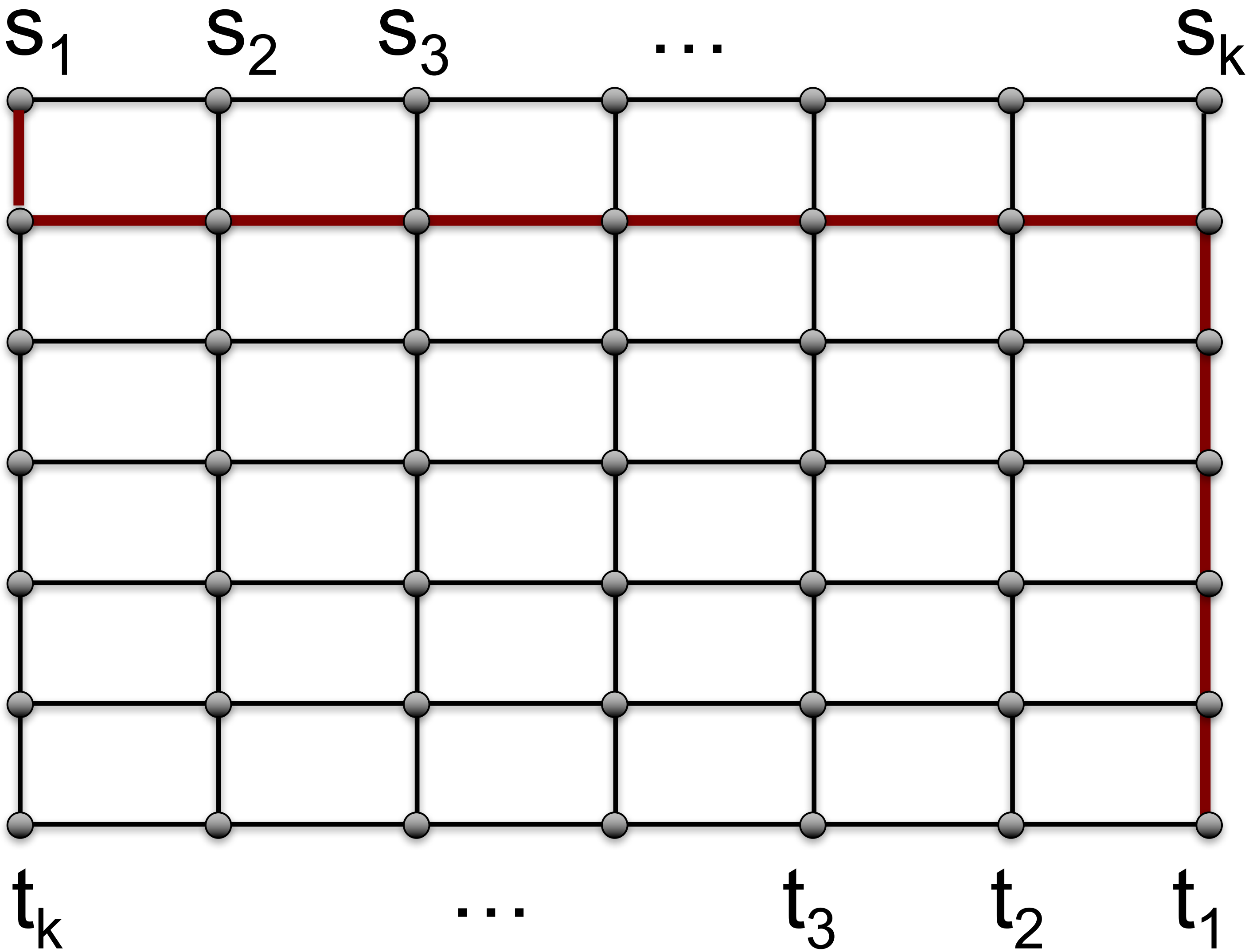
Can We Do Better?

- Not if we use the maximum multicommodity flow approach!

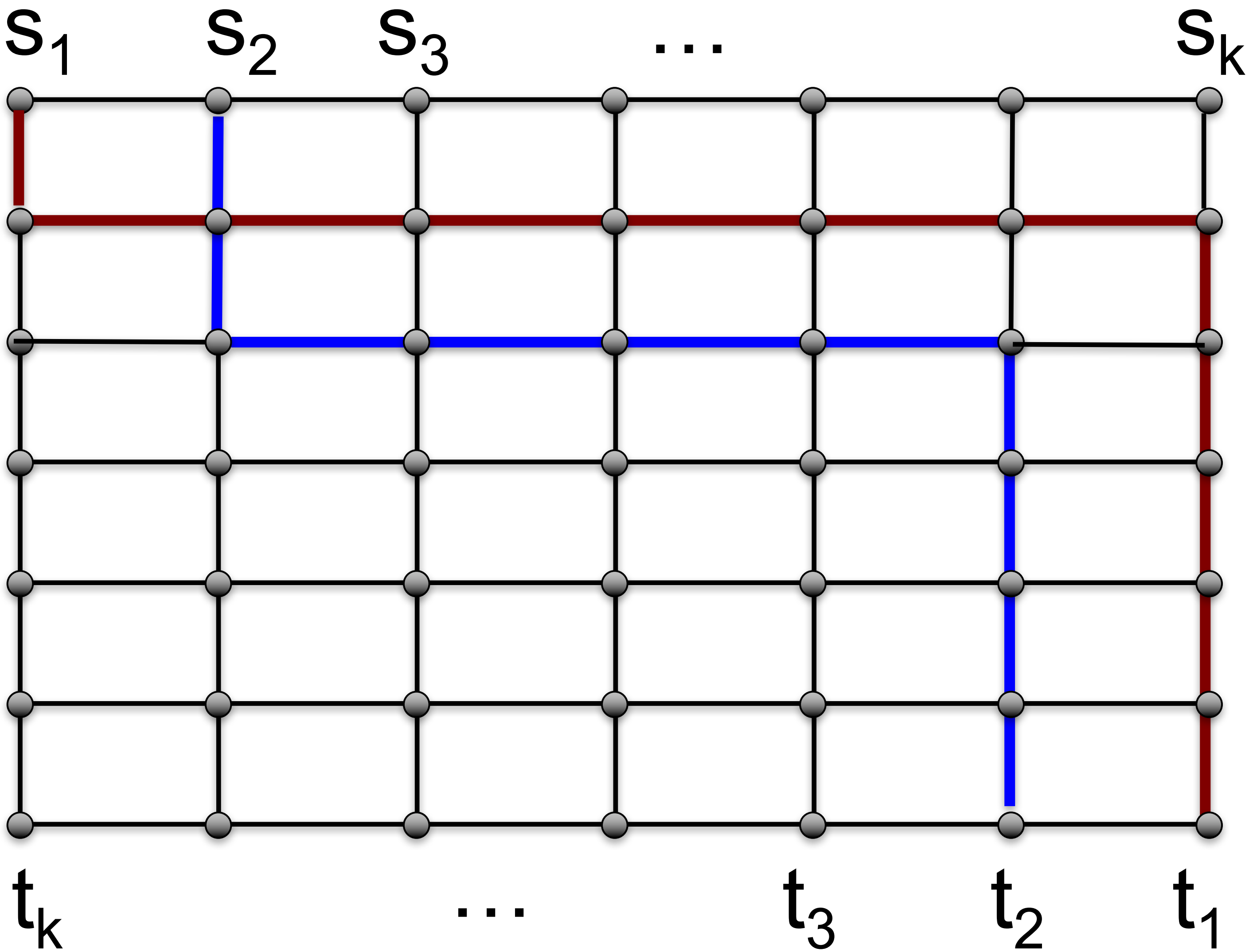
Bad Example



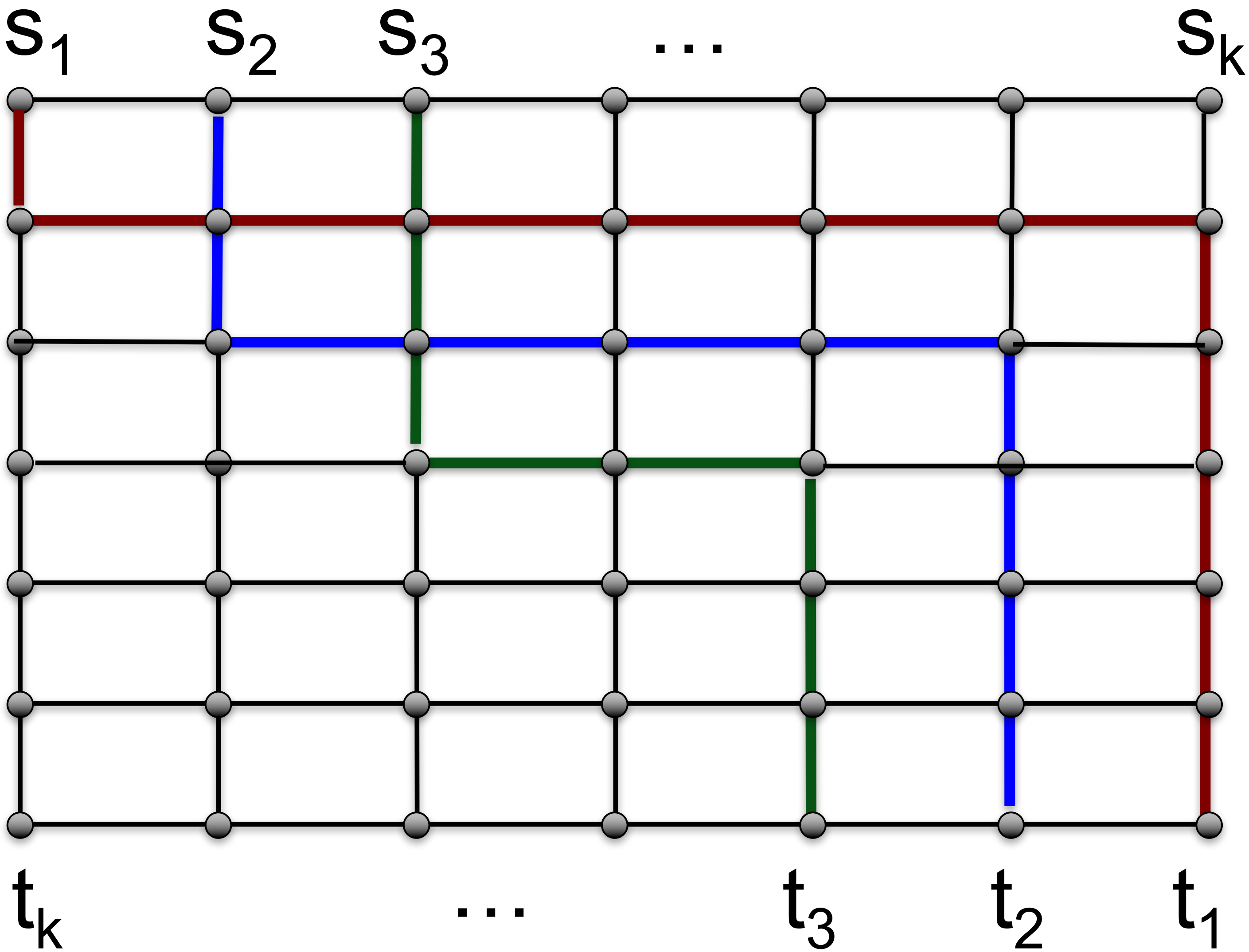
Bad Example



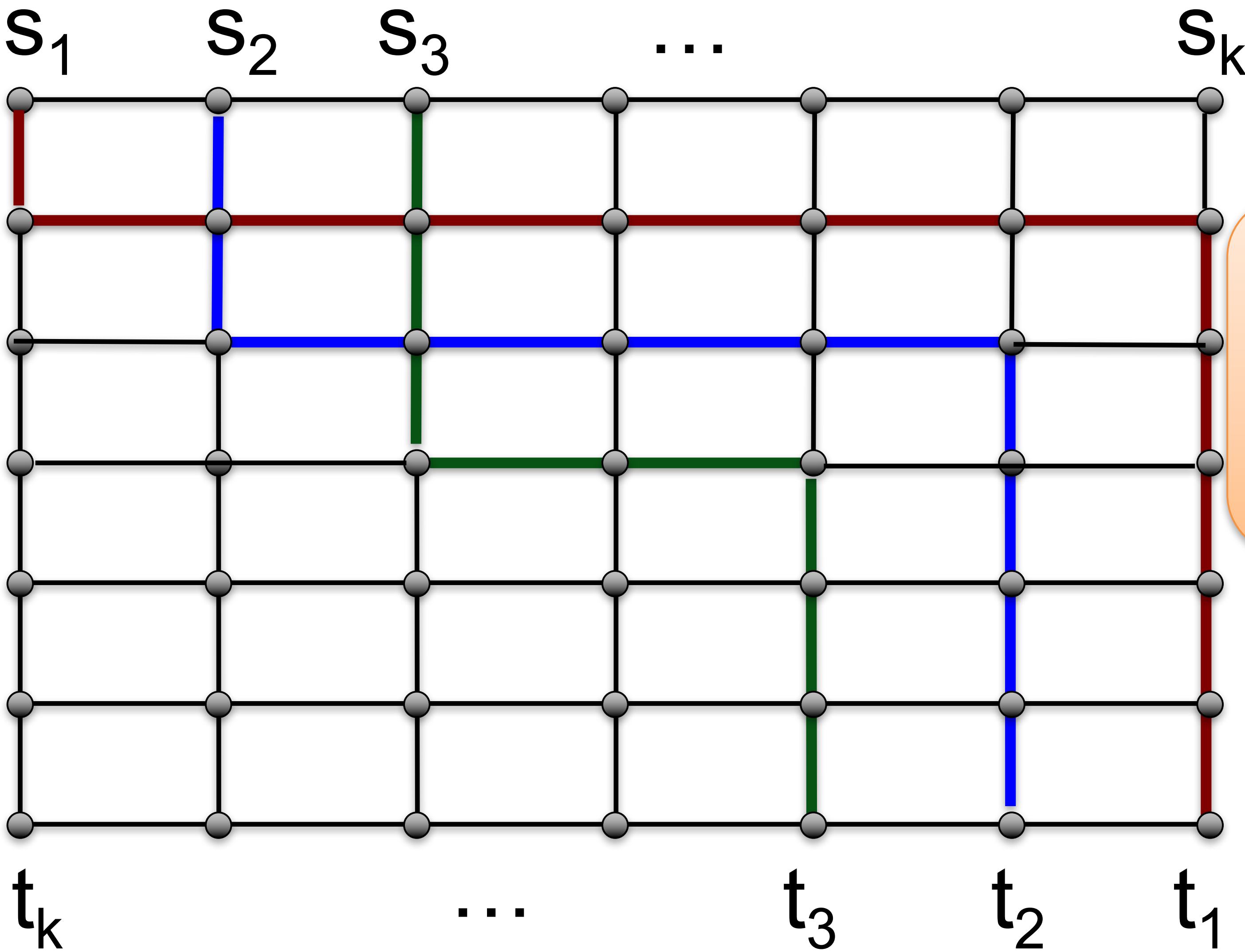
Bad Example



Bad Example



Bad Example



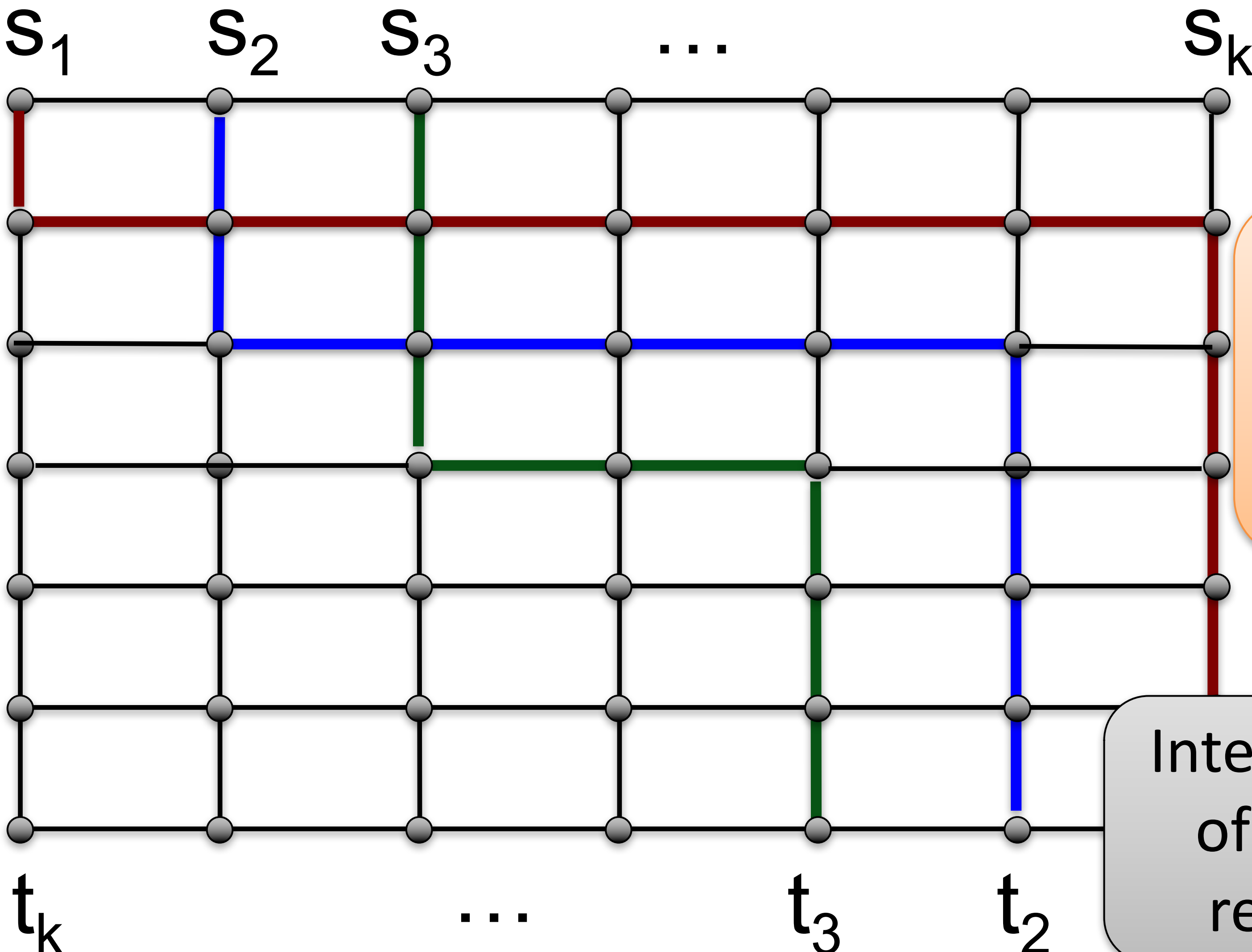
OPT_{flow}=k/3

OPT=1

gap:

$\Omega(k) = \Omega(\sqrt{n})$

Bad Example



OPT_{flow}=k/3
OPT=1
gap:
$\Omega(k) = \Omega(\sqrt{n})$

Integrality gap of the flow relaxation

Can We Do Better?

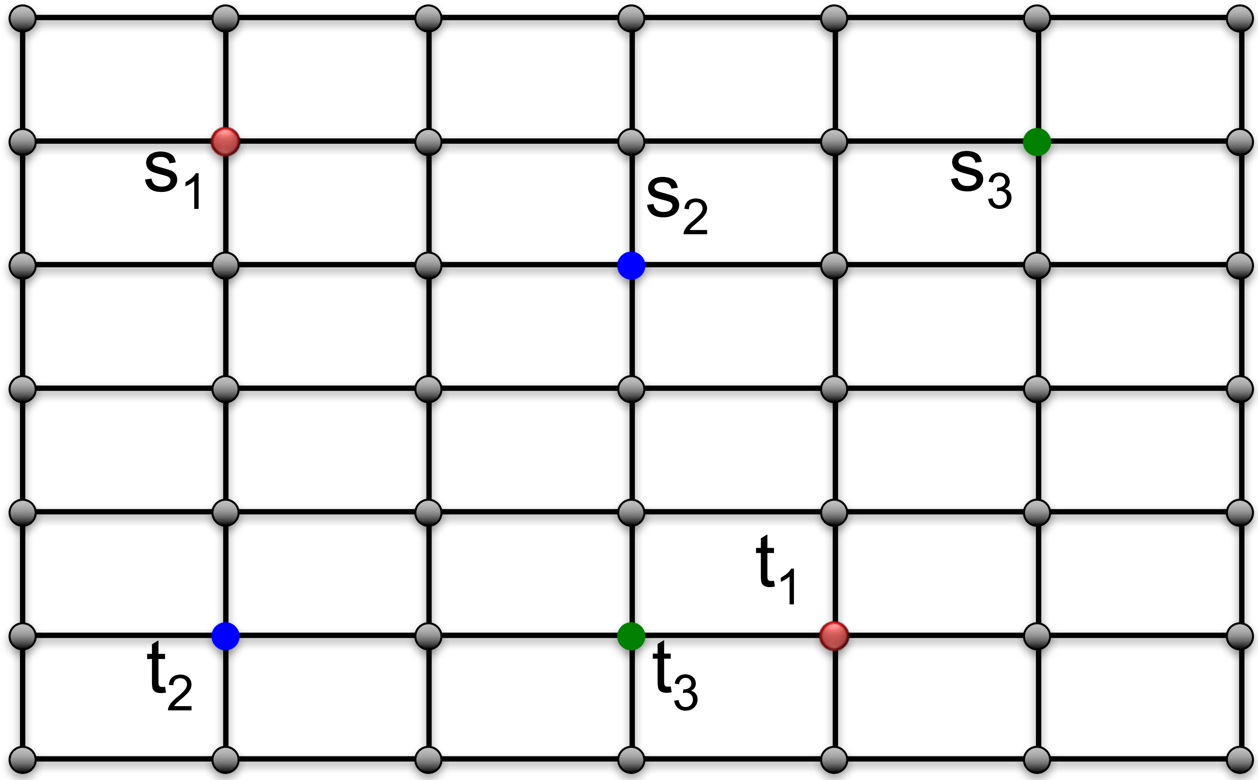
- Not if we use the maximum multicommodity flow approach!
- $\Omega(\log^{1/2-\epsilon} n)$-hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Approximation Status of NDP

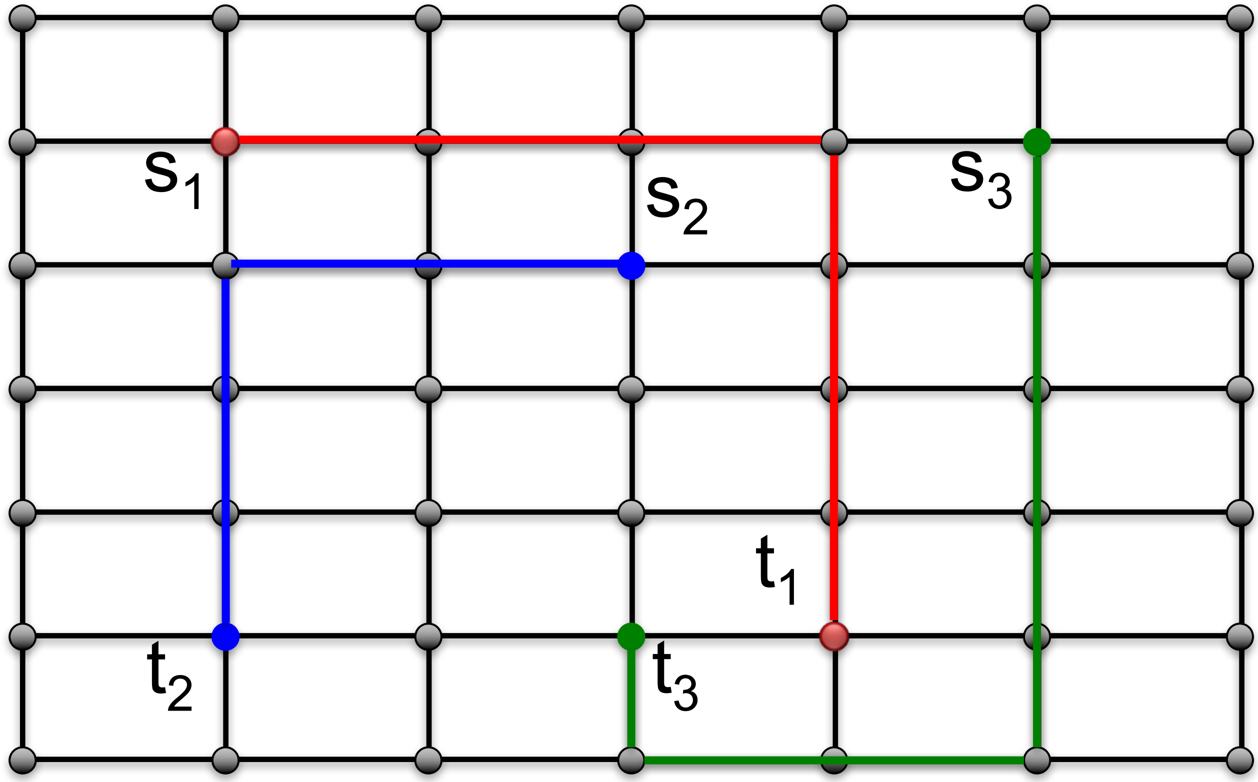
- $O(\sqrt{n})$ -approximation algorithm
  - even on planar graphs
  - even on grid graphs
- $\Omega(\log^{1/2-\epsilon} n)$ -hardness of approximation for any ϵ [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Only NP-hardness known for planar graphs and grids

NDP in Grids



NDP in Grids



Approximation Status of NDP

- $O(\sqrt{n})$ -approximation algorithm

Until recently:

- even on planar graphs
- even on grid graphs

[C, Kim '15]
$\tilde{O}(n^{1/4})$-approximation

- $\Omega(\log^{1/2-\epsilon} n)$-hardness o
  [Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Only NP-hardness known for planar graphs and grids

Approximation Status of NDP

- $O(\sqrt{n})$ -approximation alg

Until recently:

– even on planar graphs

– even on grid graphs

[C, Kim, Li '16]

$\tilde{O}(n^{9/19})$-approximation

[C, Kim '15]

$\tilde{O}(n^{1/4})$-approximation

- $\Omega(\log^{1/2-\epsilon} n)$-hardness o

[Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

Approximation Status of NDP

- $O(\sqrt{n})$ -approximation alg

Until recently:

– even on planar graphs

– even on grid graphs

[C, Kim, Li '16]

$\tilde{O}(n^{9/19})$-approximation

[C, Kim '15]

$\tilde{O}(n^{1/4})$-approximation

- $\Omega(\log^{1/2-\epsilon} n)$-hardness o

[Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

[C, Kim, Nimavat '16]

$2^{\Omega(\sqrt{\log n})}$-hardness of approximation for subgraphs of grids

Approximation Status of NDP

- $O(\sqrt{n})$ -approximation alg

Until recently:

– eve

– eve

- $\Omega(\log$ hardness o

[Andrews, Zhang '05], [Andrews, C, Guruswami, Khanna, Talwar, Zhang '10]

[C, Kim, Li '16]

$\tilde{O}(n^{9/19})$-approximation

Work in Progress:

Almost polynomial hardness for NDP in grid graphs [C, Kim, Nimavat '17]

on

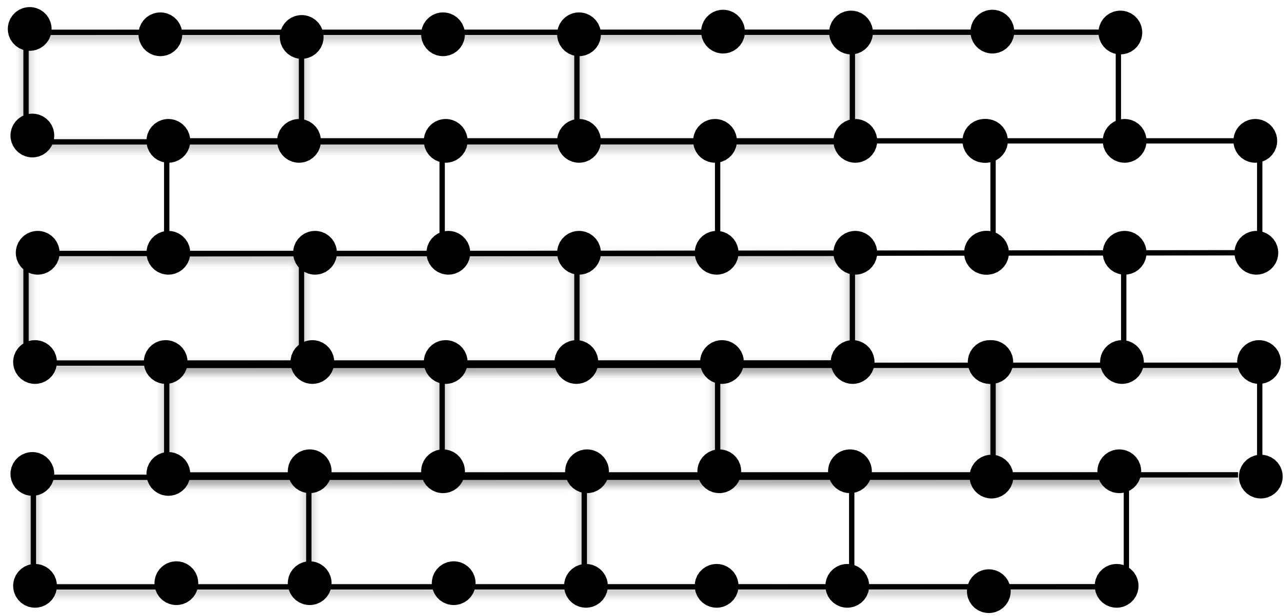
[C, Kim, Nimavat '16]

$2^{\Omega(\sqrt{\log n})}$-hardness of approximation for subgraphs of grids

Approximation Status of EDP

- $O(\sqrt{n})$-approximation algorithm [Chekuri, Khanna, Shepherd '06]
- $2^{\Omega(\sqrt{\log n})}$ -hardness of approximation even for subgraphs of wall graphs [C, Kim, Nimavat '16]

A Wall



Approximation Status of EDP

- $O(\sqrt{n})$-approximation algorithm [Chekuri, Khanna, Shepherd '06]
- $2^{\Omega(\sqrt{\log n})}$-hardness of approximation even for subgraphs of wall graphs [C, Kim, Nimavat '16]
- Work in progress: almost polynomial hardness for EDP on wall graphs [C, Kim, Nimavat '17]

Summary so Far

EDP and NDP do not have reasonable approximation algorithms, even on planar graphs

What if we allow some congestion?

EDP/NDP with Congestion

An α-approximation algorithm with congestion c routes OPT/α demand pairs with congestion at most c.

up to c paths can share an edge or a vertex

EDP/NDP with Congestion

An α-approximation algorithm with congestion c routes OPT/α demand pairs with congestion at most c.

optimum number of pairs with no congestion allowed

EDP with Congestion

- Congestion O(log n/log log n): constant approximation [Raghavan, Thompson '87]
- Congestion c: $O(n^{1/c})$-approximation [Azar, Regev '01], [Baveja, Srinivasan '00], [Kolliopoulos, Stein '04]
- Congestion poly(log log n): polylog(n)-approx [Andrews '10]
- Congestion 2: $O(n^{3/7})$-approximation [Kawarabayashi, Kobayashi '11]
- Congestion 14: polylog(k)-approximation [C, '11]
- Congestion 2: polylog(k)-approximation [C, Li '12]
- polylog(k)-approximation for NDP with congestion 2 [Chekuri, Ene '12], [Chekuri, C '16]

EDP with Congestion

- Congestion O(log n/log log n): constant approximation [Raghava
- Congestion c: $O(n^{1/c})$- [Baveja, Srinivasan ’00], [Kolli
- Congestion poly(log log n): polylog(n)-approx [Andrews ‘10]
- Congestion 2: $O(n^{3/7})$-approx Kobayashi ’11]
- Congestion 14: polylog(k)-approximation [C, ‘11]
- Congestion 2: polylog(k)-approximation [C, Li ’12]
- polylog(k)-approximation for NDP with congestion 2 [Chekuri, Ene ’12], [Chekuri, C ‘16]

All these results are based on the multicommodity flow relaxation

“Tight” due to known hardness results

EDP with Congestion

- Congestion O(log n/log log n): constant approximation [Raghavan, Thompson '87]
- Congestion c: $O(n^{1/c})$-approximation [Azar, Regev '01], [Baveja, Srinivasan '00], [Kolliopoulos, Stein '04]
- Congestion poly(log log n): polylog(n)-approx [Andrews '10]
- Congestion 2: $O(n^{3/7})$- … rabayashi, Kobayashi '11]

Structural results about graphs

- Congestion 14: polylog(k)-approximation [C, '11]
- … ylog(k)-approximation [C, Li '12]
- … imation for NDP with congestion 2 [Chekuri, Ene '12], [Chekuri, C '16]

new results in graph theory!

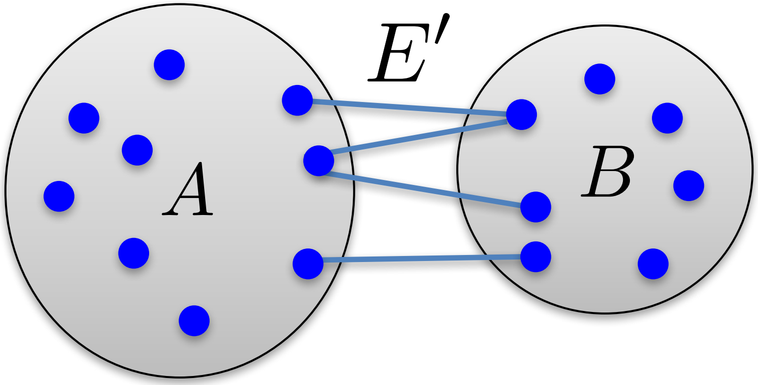
EDP with Congestion

- Congestion O(log n/log log n): constant approximation [Raghavan, Thompson '87]
- Congestion c: $O(n^{1/c})$-approximation [Azar, Regev '01], [Baveja, Srinivasan '00], [Kolliopoulos, Stein '04]
- Congestion poly(log log n): polylog(n)-approx [Andrews '10]
- Congestion 2: $O(n^{3/7})$-approximation [Kawarabayashi, Kobayashi '11]
- Congestion 14: polylog(k)-approximation [C, '11]
- Congestion 2: polylog(k)-approximation [C, Li '12]
- polylog(k)-approximation for NDP with congestion 2 [Chekuri, Ene '12], [Chekuri, C '16]

Edge-Disjoint Paths with Constant Congestion

EDP on Expanders

$$|E'| \geq \frac{\min\{|A|, |B|\}}{2}$$



In a strong enough expander, if the set of demand pairs is not too large, can route almost all of them on Node-Disjoint Paths!

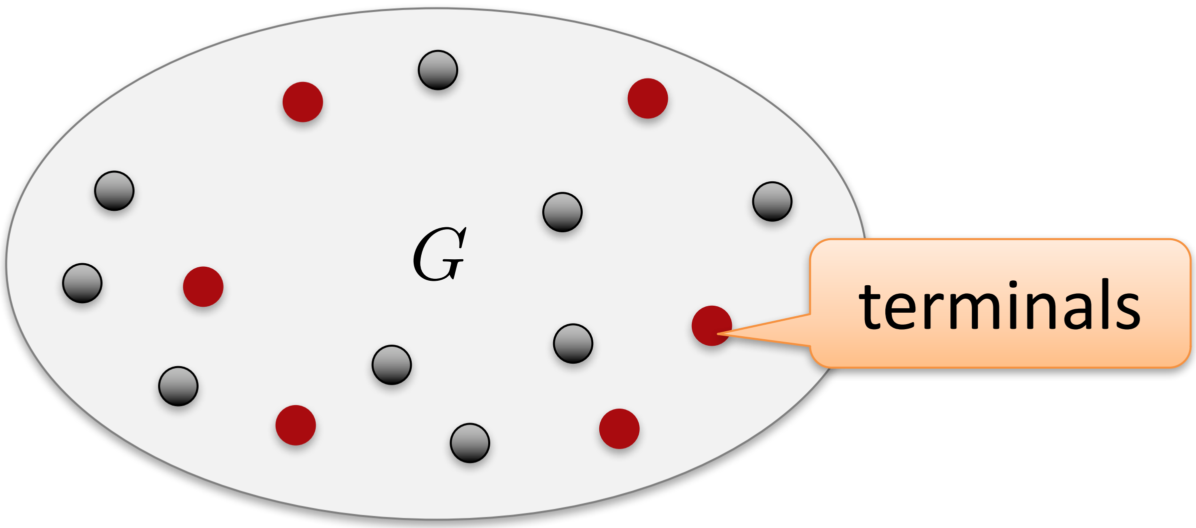
Main Idea: Exploit Algorithms for Expanders!

But our graph is nothing like an expander

Find expander-like structure in the graph and use it for routing!

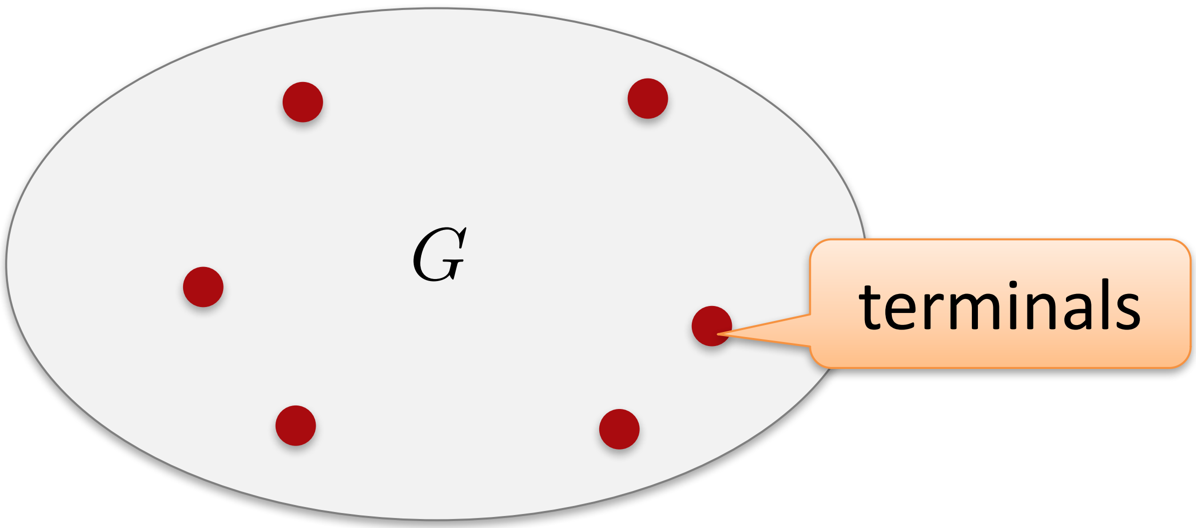
Well-Linkedness

[Robertson,Seymour], [Chekuri, Khanna, Shepherd], [Raecke]



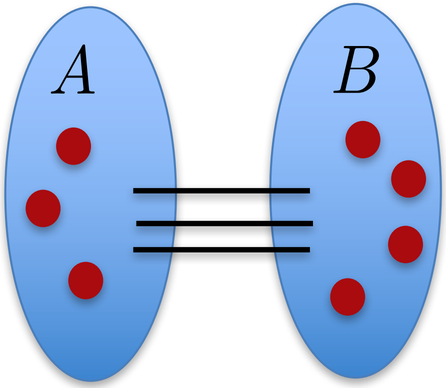
Well-Linkedness

[Robertson,Seymour], [Chekuri, Khanna, Shepherd], [Raecke]



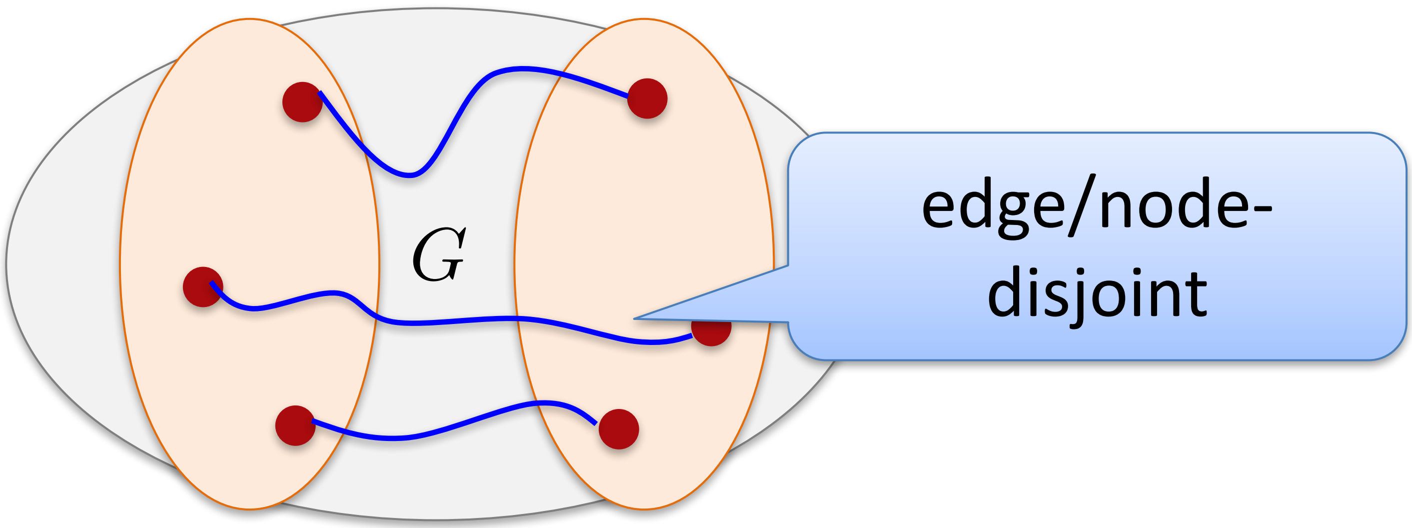
Set T of terminals is well-linked in G, iff for any partition (A,B) of V(G),

$$|E(A,B)| \geq \min\{|A \cap T|, |B \cap T|\}$$



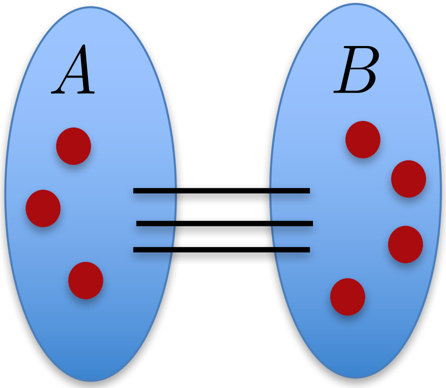
Well-Linkedness

[Robertson,Seymour], [Chekuri, Khanna, Shepherd], [Raecke]



Set T of terminals is well-linked in G, iff for any partition (A,B) of V(G),

$$|E(A,B)| \geq \min\{|A \cap T|, |B \cap T|\}$$



EDP: Well-Linked Instances

- Terminals: vertices participating in the demand pairs

- An instance is well-linked iff the set of terminals is well-linked in G.

Theorem [Chekuri, Khanna Shepherd '04]: an α-approximation algorithm on well-linked instances gives an $O(\alpha \log^2 k)$-approximation on any instance.

EDP: Well-Linked Instances

- Terminals: vertices participating in the demand pairs

- An instance is well-linked iff t
terminals is well-linked in G.

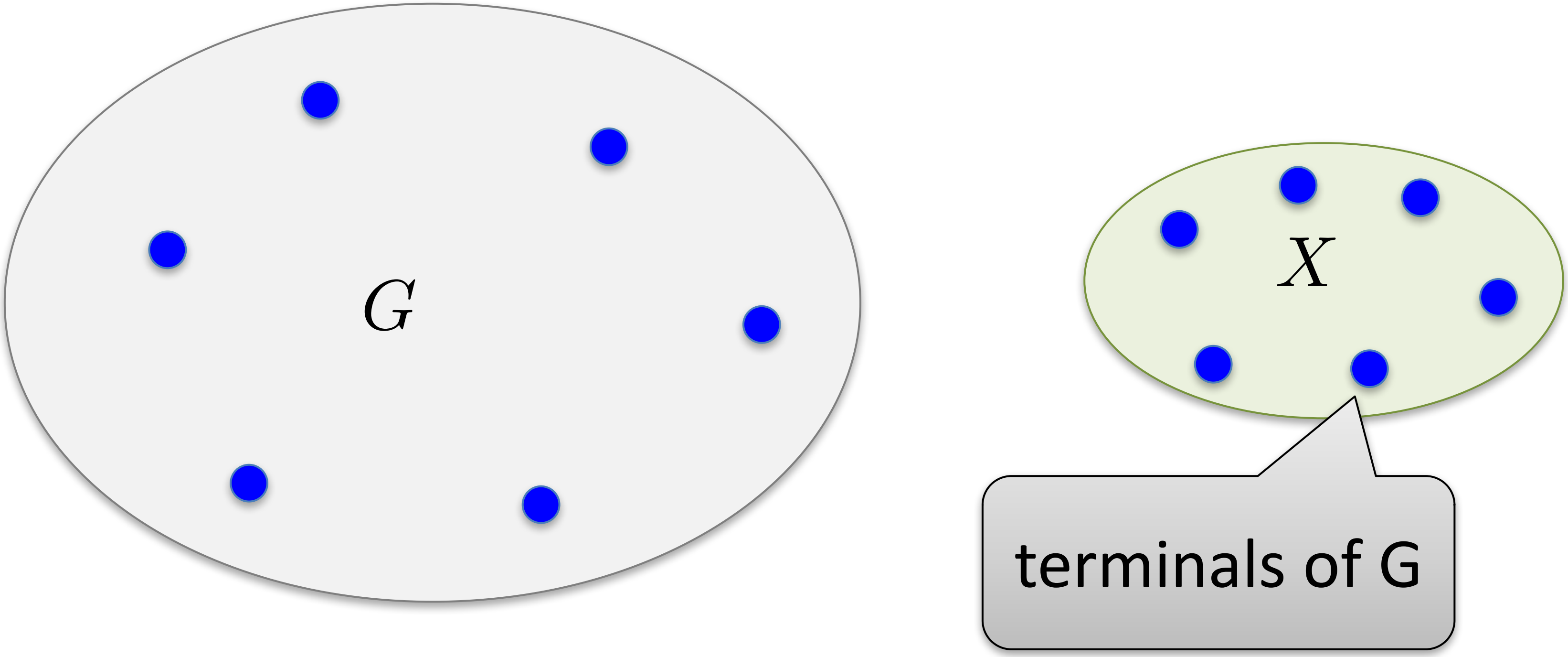
Only true if the algorithm rounds the flow relaxation

Theorem [Chekuri, Khanna Shepherd '04]: an α-approximation algorithm on well-linked instances gives an $O(\alpha \log^2 k)$-approximation on any instance.

Main Idea

[Chekuri, Khanna, Shepherd], [Rao, Zhou]

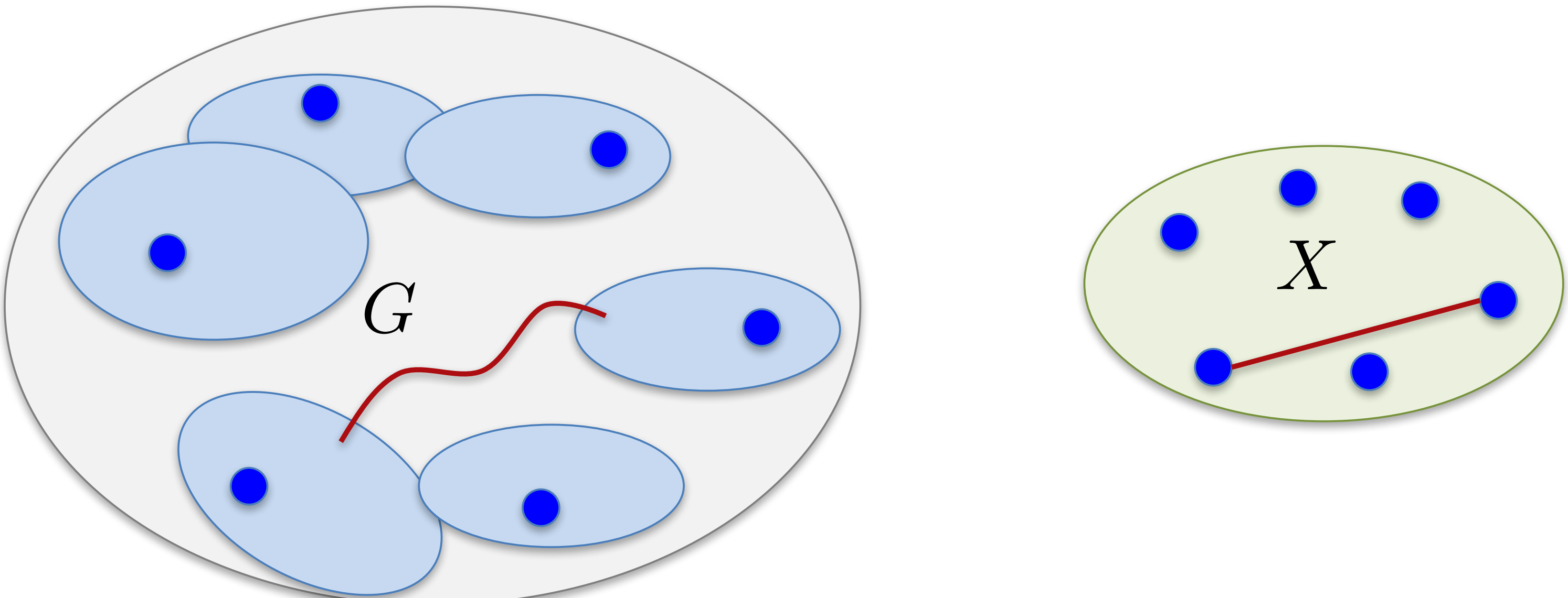
Embed an expander over the terminals into G!



Main Idea

[Chekuri, Khanna, Shepherd], [Rao, Zhou]

Embed an expander over the terminals into G!

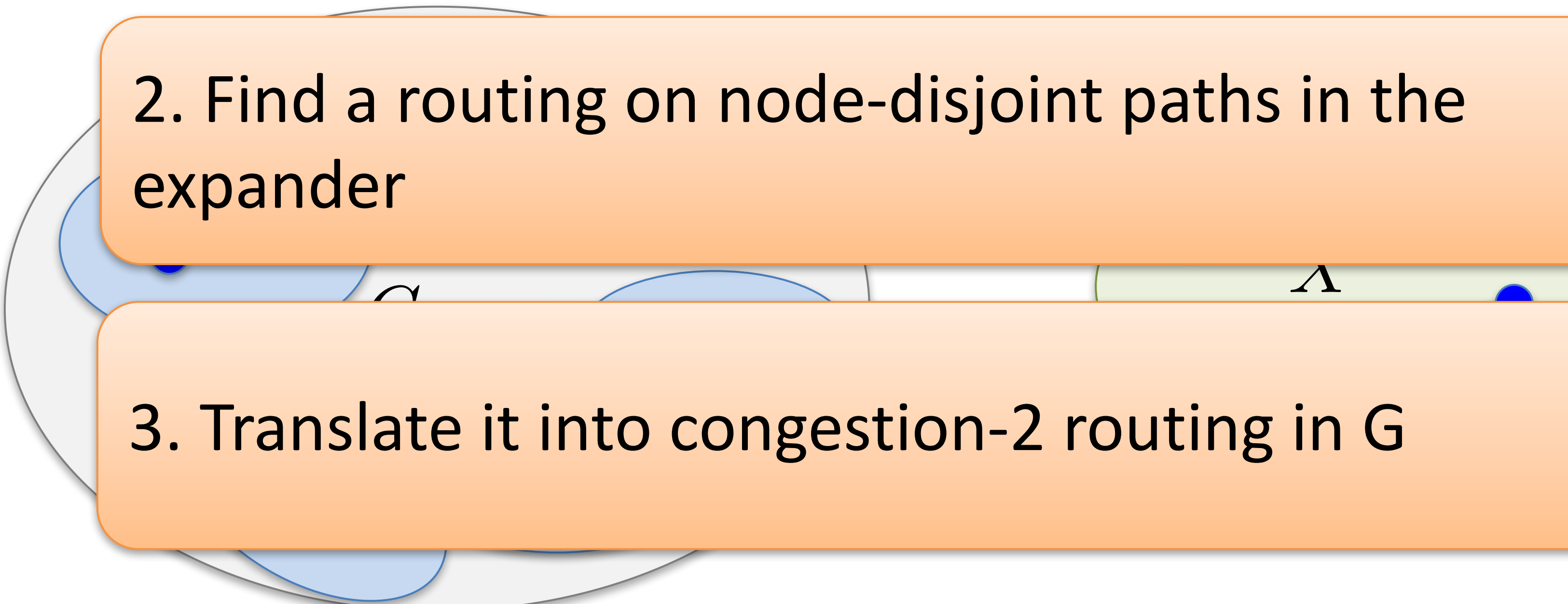


An edge of G may belong to at most 2 clusters/paths

Main Idea

[Chekuri, Khanna, Shepherd], [Rao, Zhou]

1. Embed an expander over the terminals into G

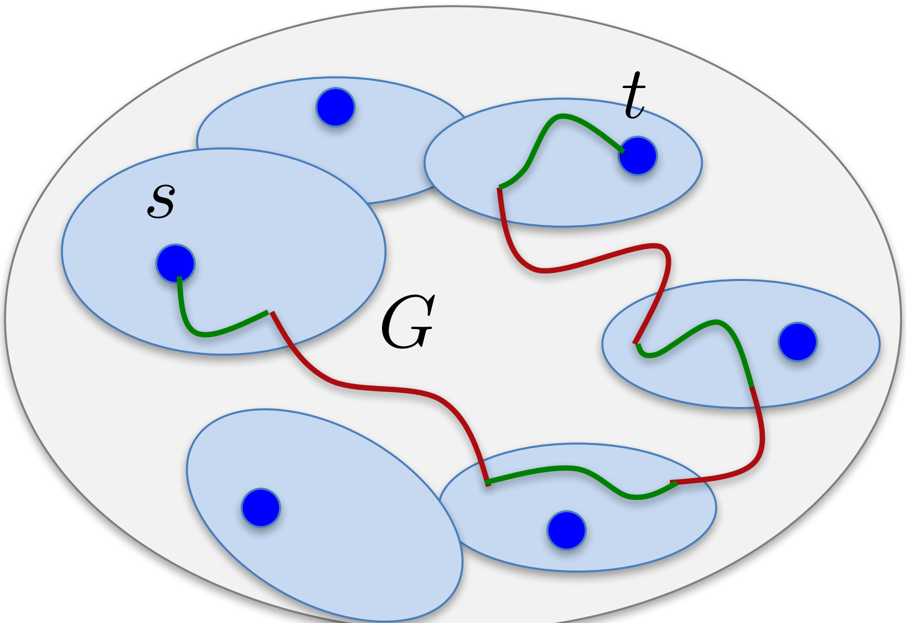


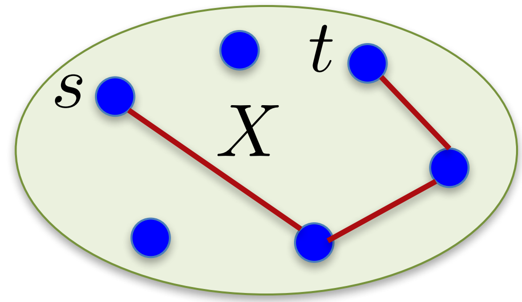
2. Find a routing on node-disjoint paths in the expander

3. Translate it into congestion-2 routing in G

An edge of G may belong to at most 2 clusters/paths

Embedding an Expander into G





Routing on vertex-disjoint paths in X gives a good routing in G!

Main Idea

1. Embed an expander over the terminals into G

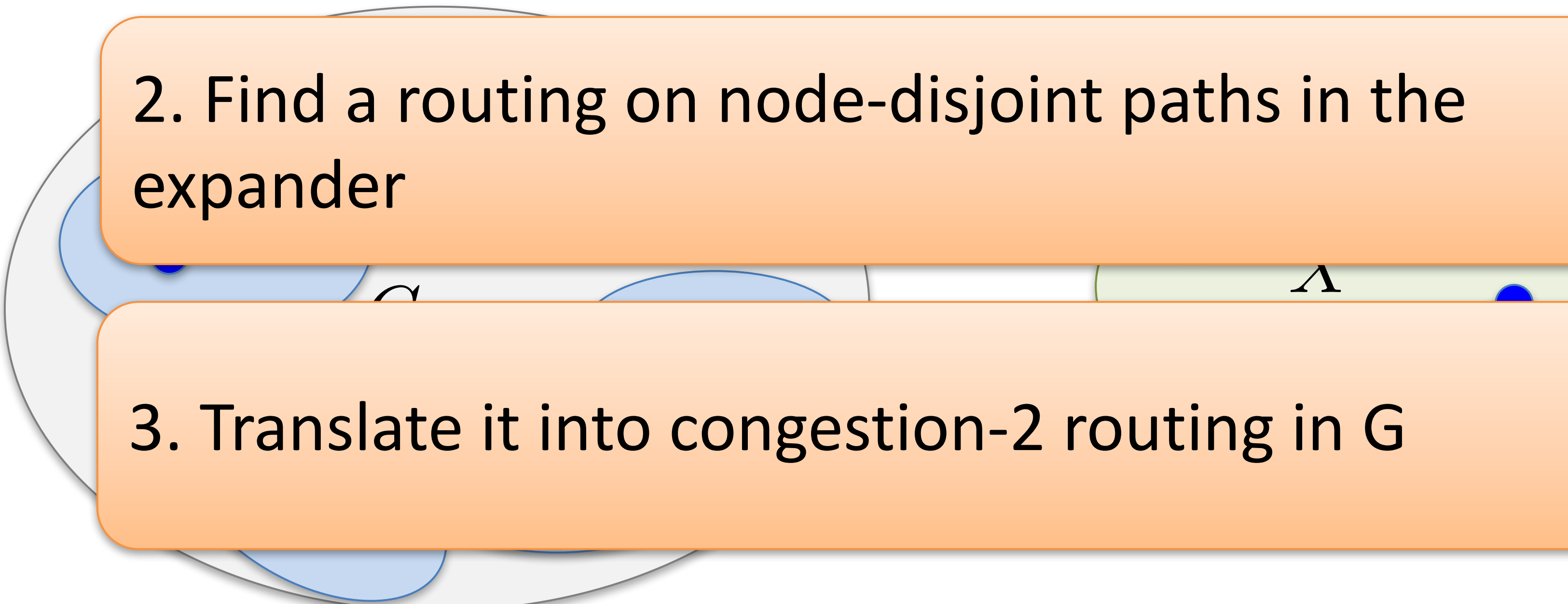
2. Find a routing on node-disjoint paths in the expander

3. Translate it into congestion-2 routing in G

An edge of G may belong to at most 2 clusters/paths

Main Idea

1. Embed an expander over the terminals into G



2. Find a routing on node-disjoint paths in the expander

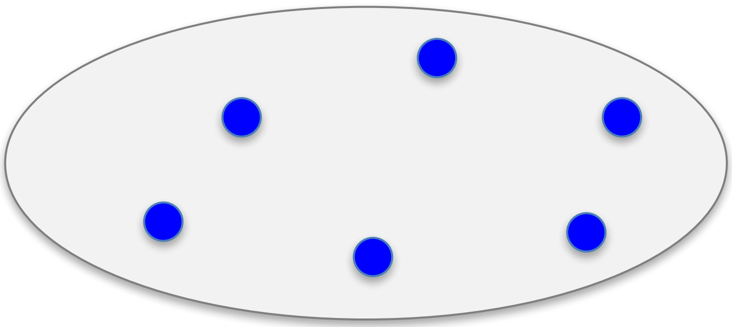
3. Translate it into congestion-2 routing in G

An edge of G may belong to at most 2 clusters/paths

Cut-Matching Game [Khandekar, Rao, Vazirani '06]

Cut Player: wants to build an expander

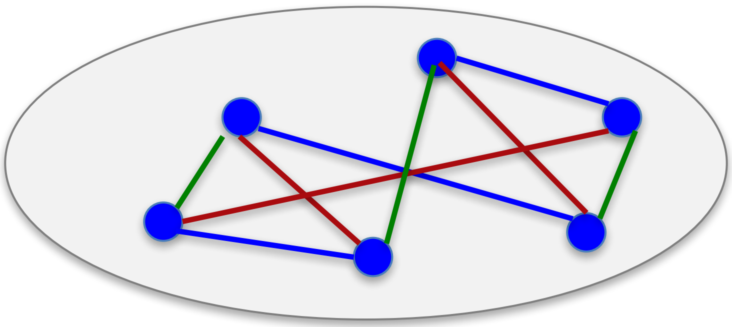
Matching Player: wants to delay its construction



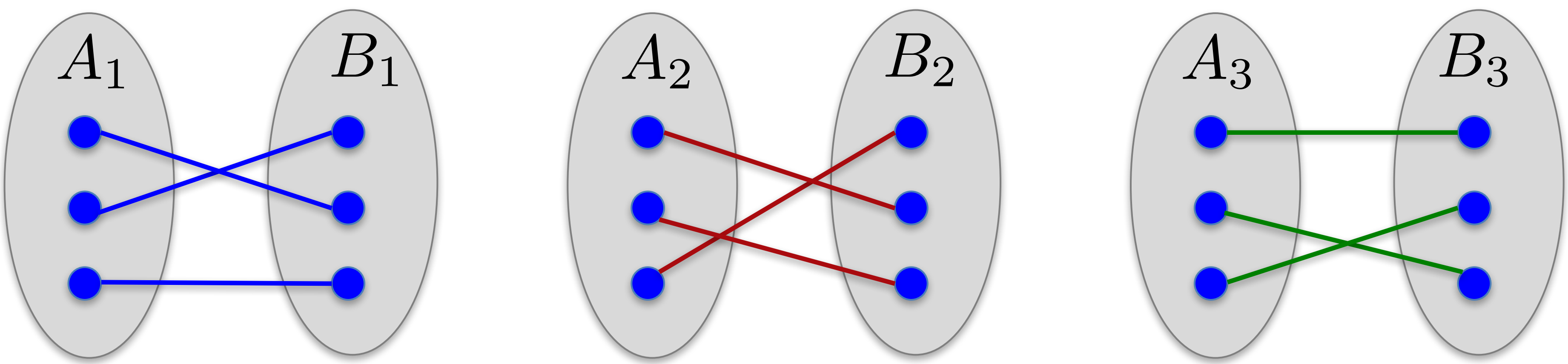
Cut-Matching Game [Khandekar, Rao, Vazirani '06]

Cut Player: wants to build an expander

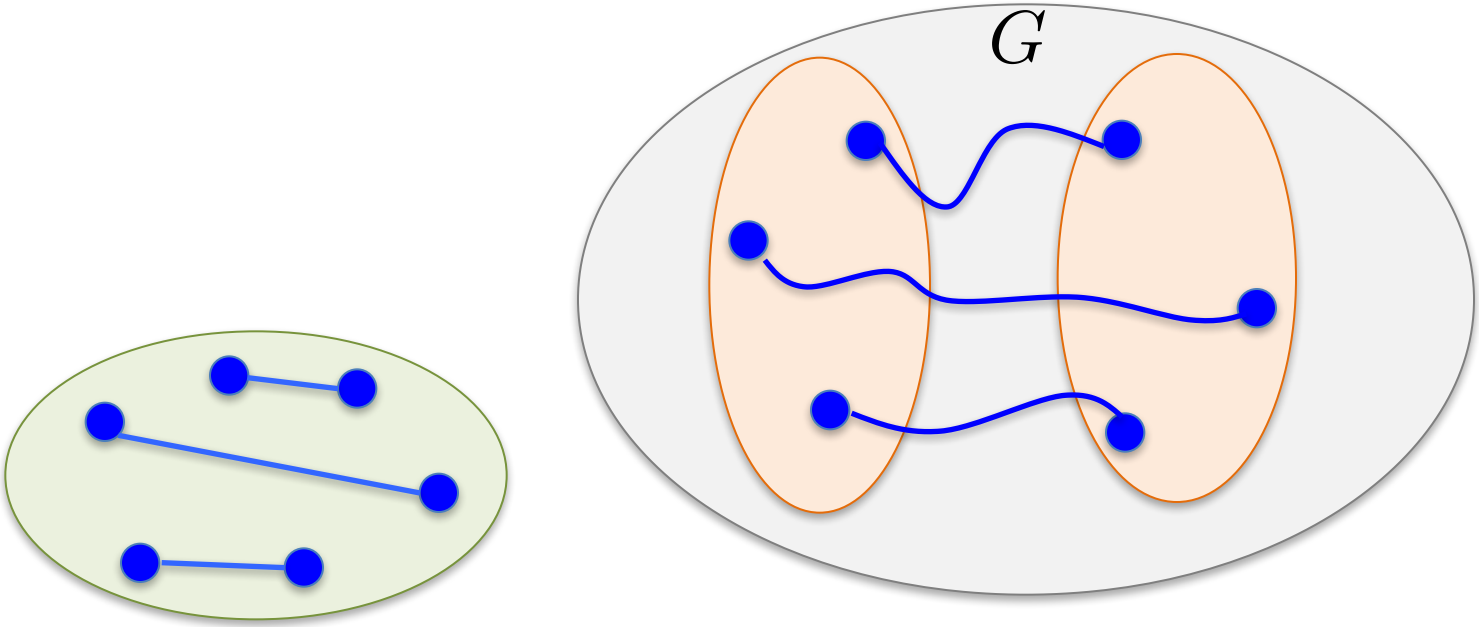
Matching Player: wants to delay its construction



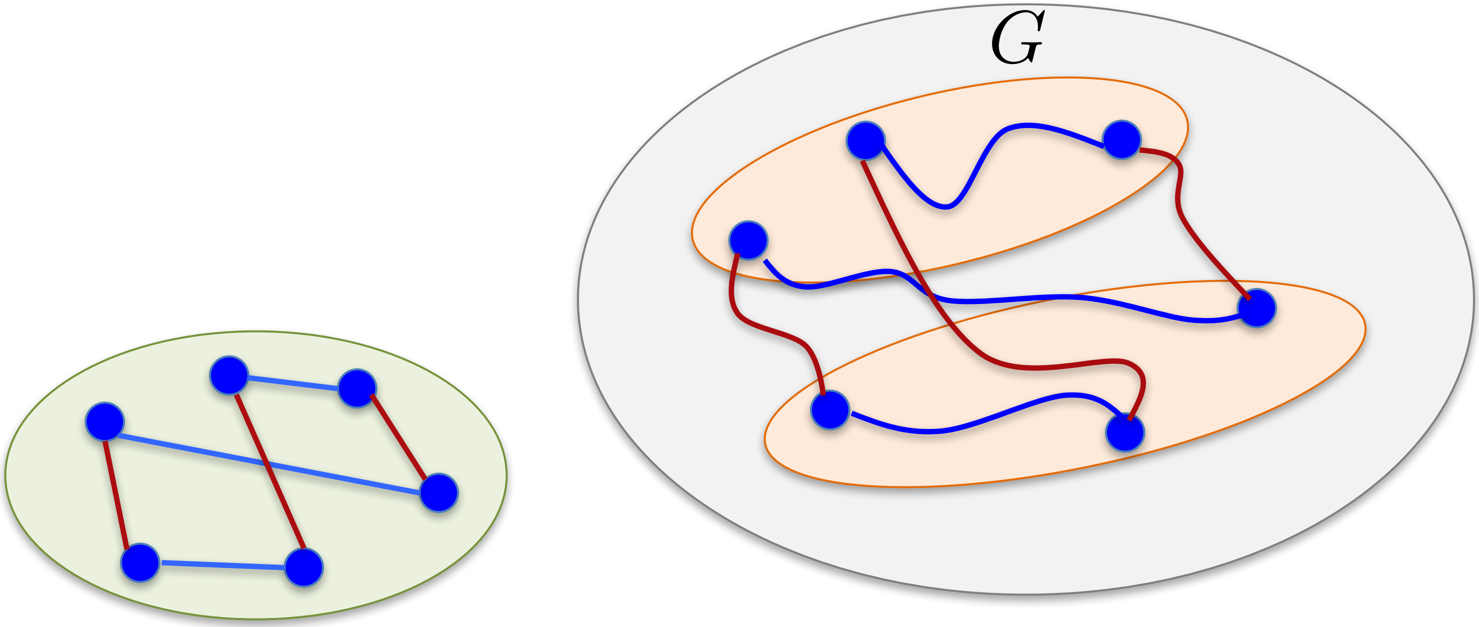
There is a strategy for cut player, s.t. after O(log^{2}n) iterations, we get an expander!



Embedding Expander into Graph



Embedding Expander into Graph



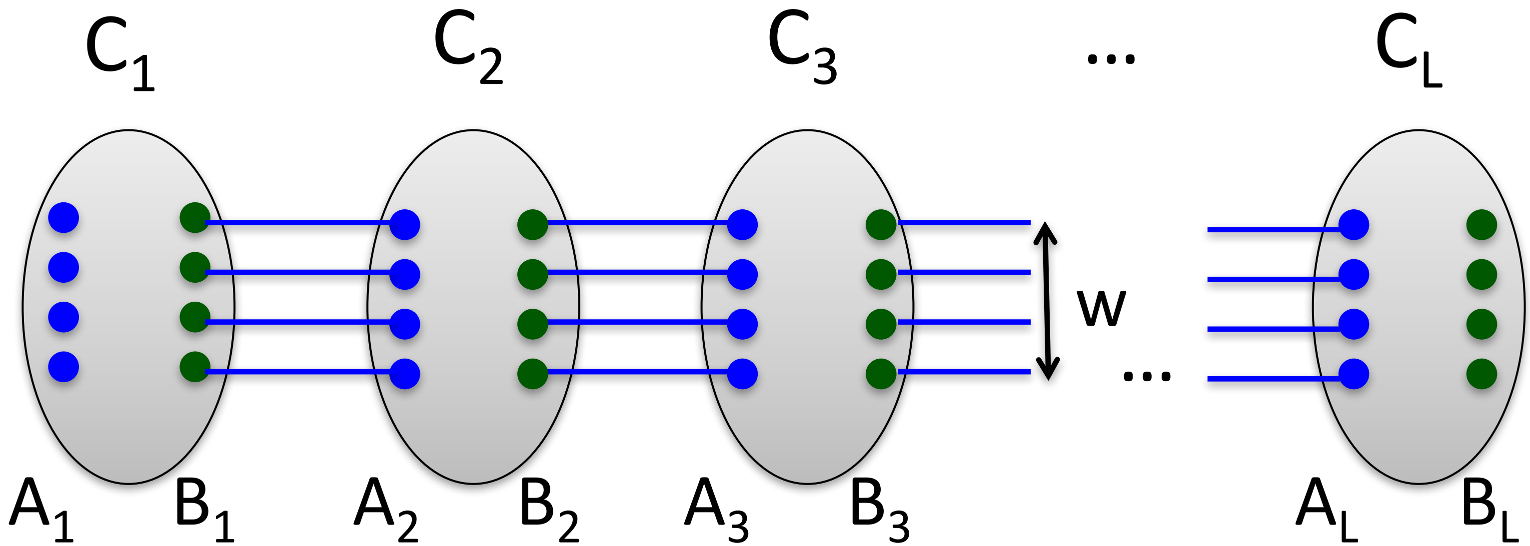
After $O(\log^2 k)$ iterations, we get an expander embedded into G.

Problem: congestion $\Omega(\log^2 k)$

Path-of-Sets System

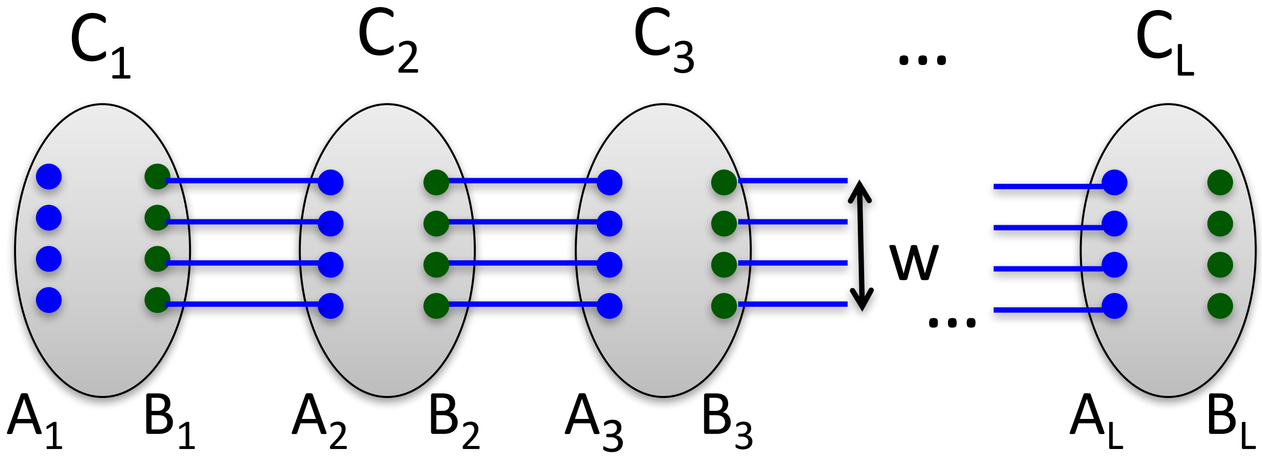
A Path-of-Sets System

width w
length L



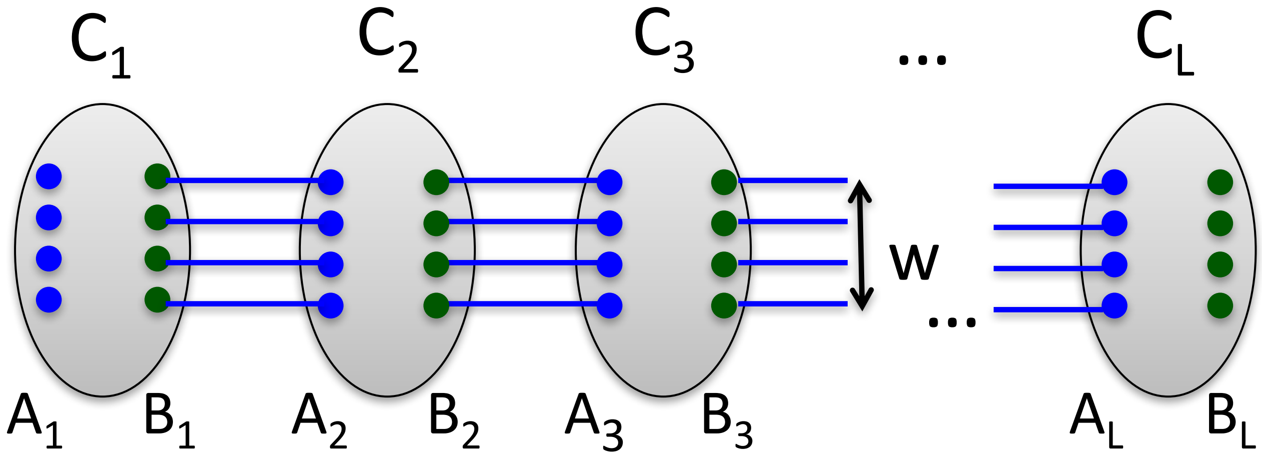
- L disjoint connected clusters
- Two disjoint sets A_i, B_i of w vertices in each cluster C_i
- $A_i \cup B_i$ is well-linked in C_i
- For all i, set P_i of w disjoint paths connecting B_i to A_{i+1}
- All paths are disjoint from each other and internally disjoint from clusters

From Well-Linkedness to Path-of-Sets



Theorem [C, '11], [C, Li '12], [Chekuri, C '13]:
Suppose G has a set of k well-linked vertices.
Then we can efficiently construct a path-of-sets system in G with parameters L and w, if: $w \cdot L^{48} < \tilde{O}(k)$

From Well-Linkedness to Path-of-Sets



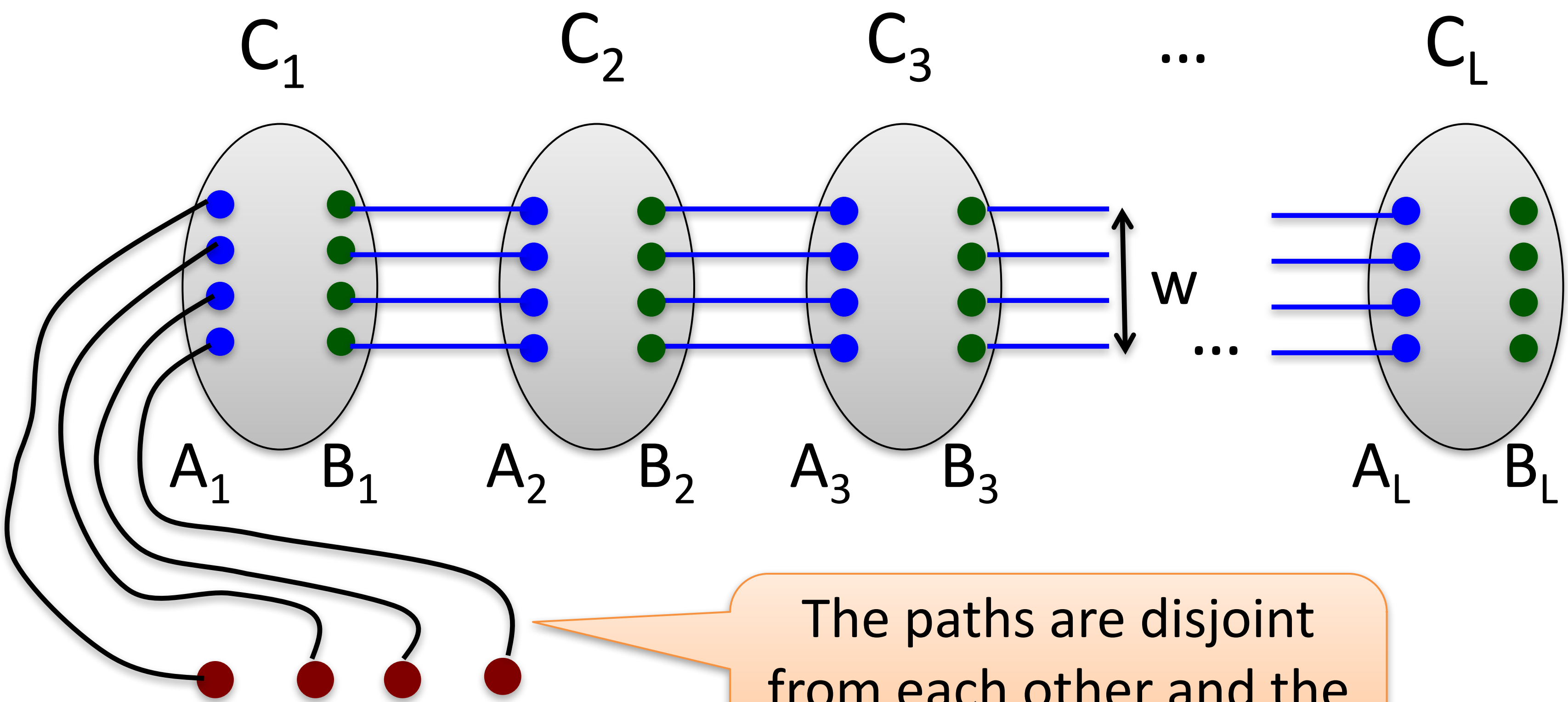
We'll use:
$L=O(\log^2 k)$
$w=k/\text{polylog } k$

Theorem [C, '11], [C, Li '12], [Chekuri, C '13]:
Suppose G has a set of k well-linked vertices.
Then we can efficiently construct a path-of-sets system in G with parameters L and w, if: $w \cdot L^{48} < \tilde{O}(k)$

Extras:

- Can connect w terminals to A_1 by disjoint paths
- Can make sure they form demand pairs!

From Well-Linkedness to Path-of-Sets

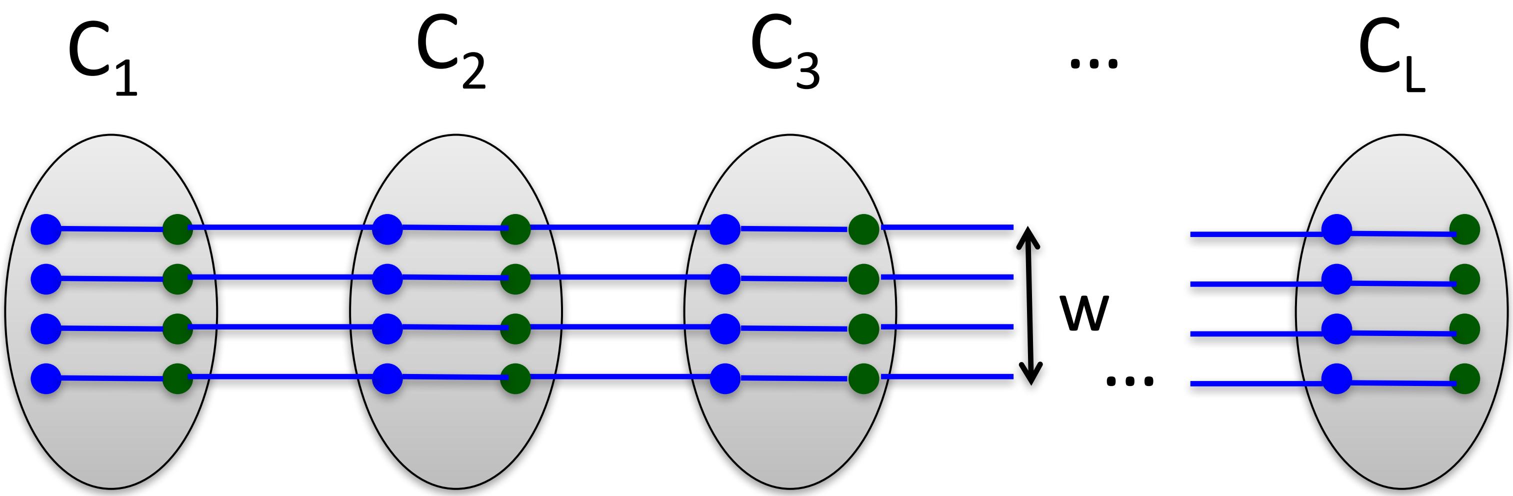


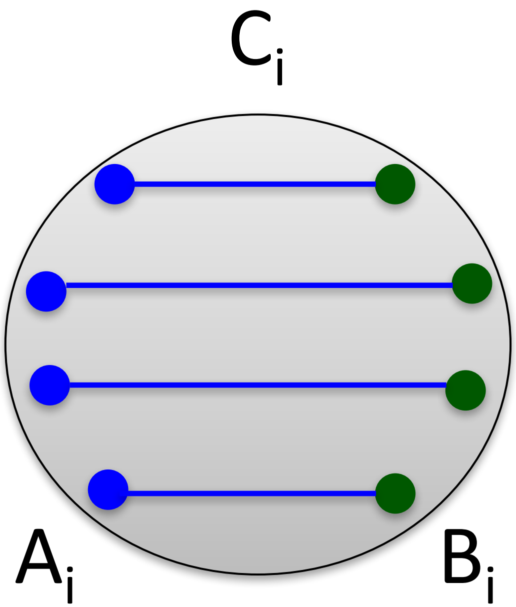
The paths are disjoint from each other and the PoS system

The terminals form demand pairs

Given the PoS, can embed an expander!

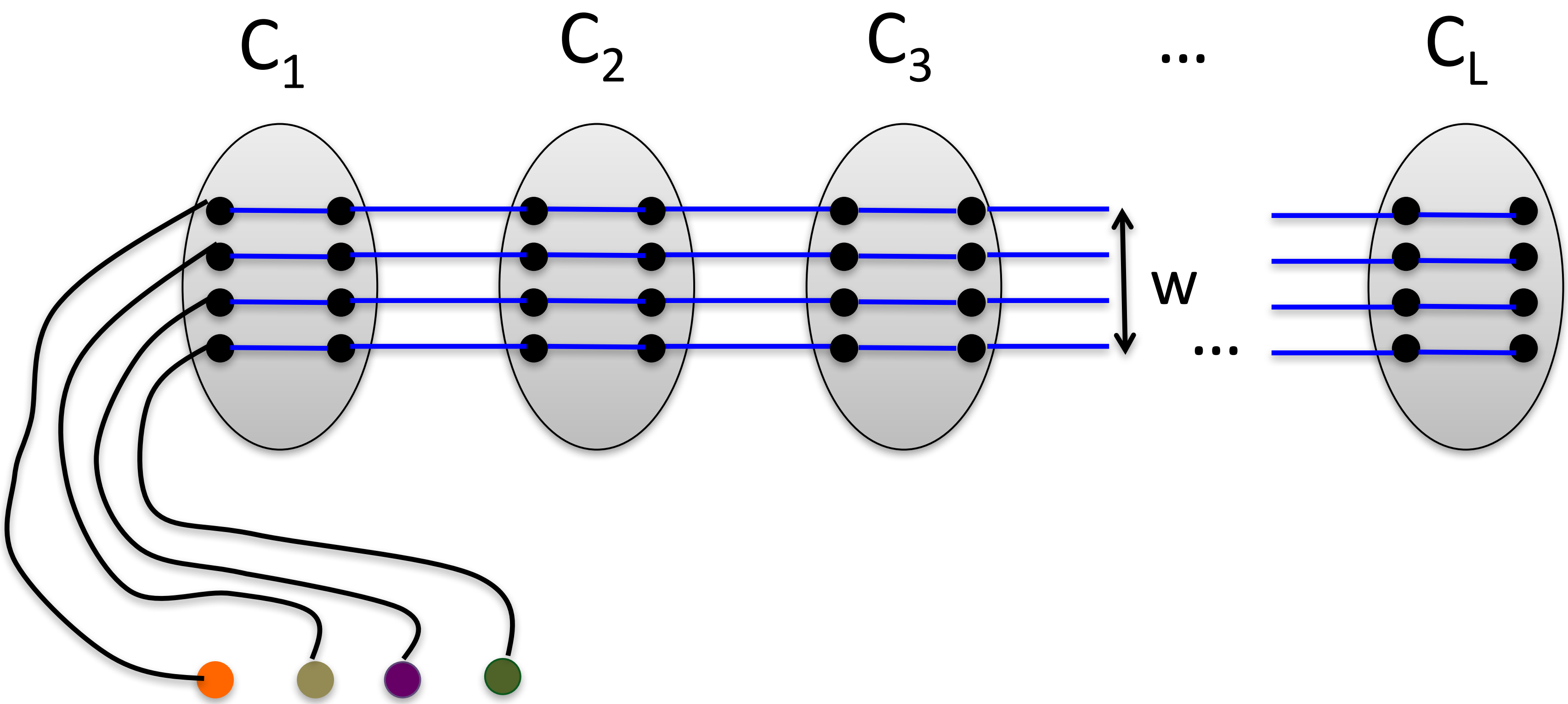
Embedding the Expander

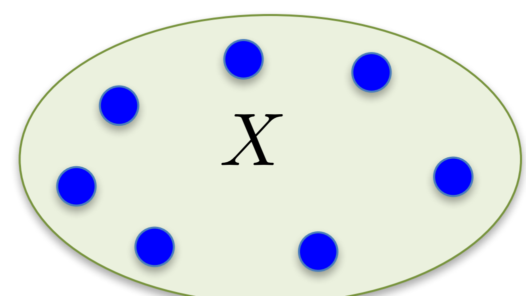




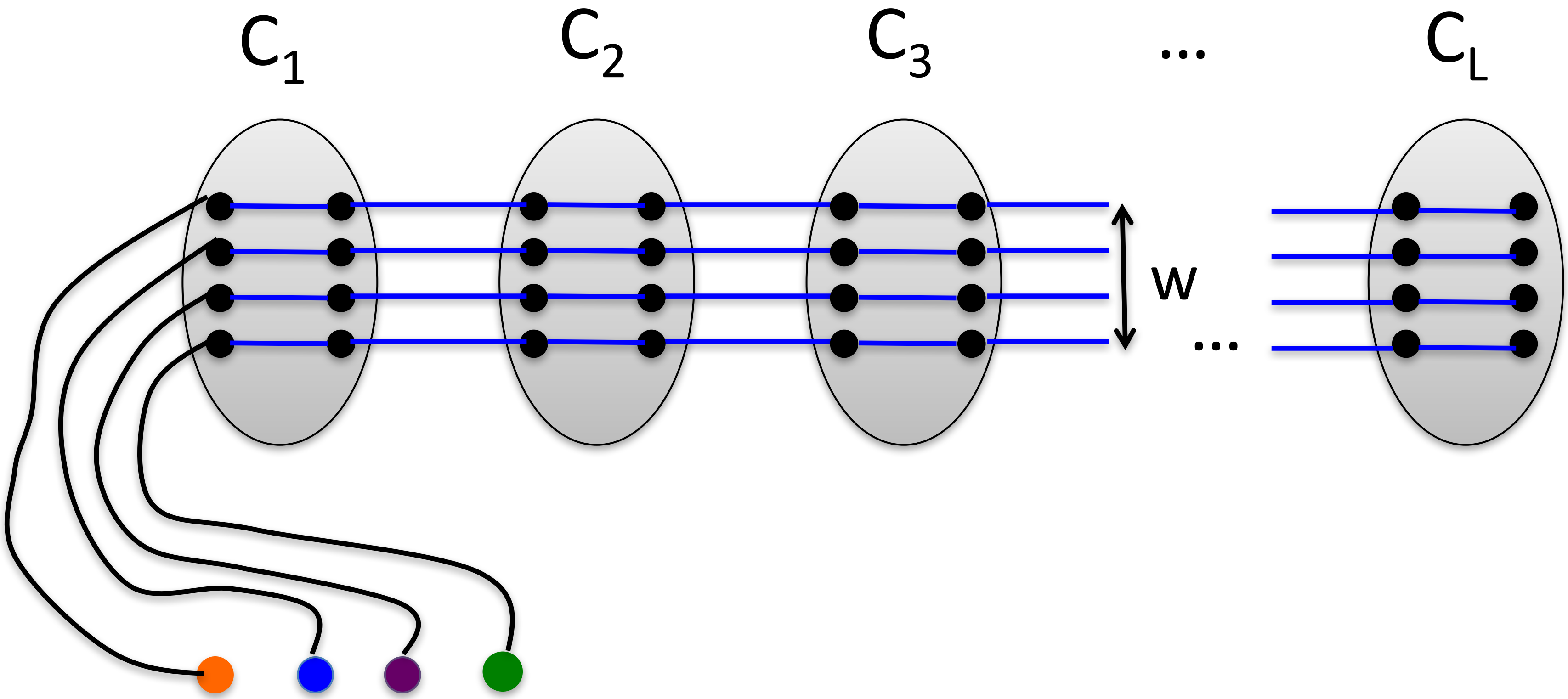
$A_i \cup B_i$ is well-linked inside C_i

Embedding the Expander

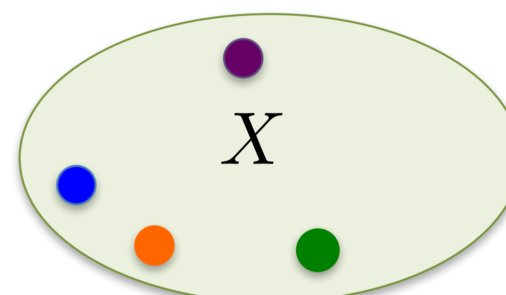




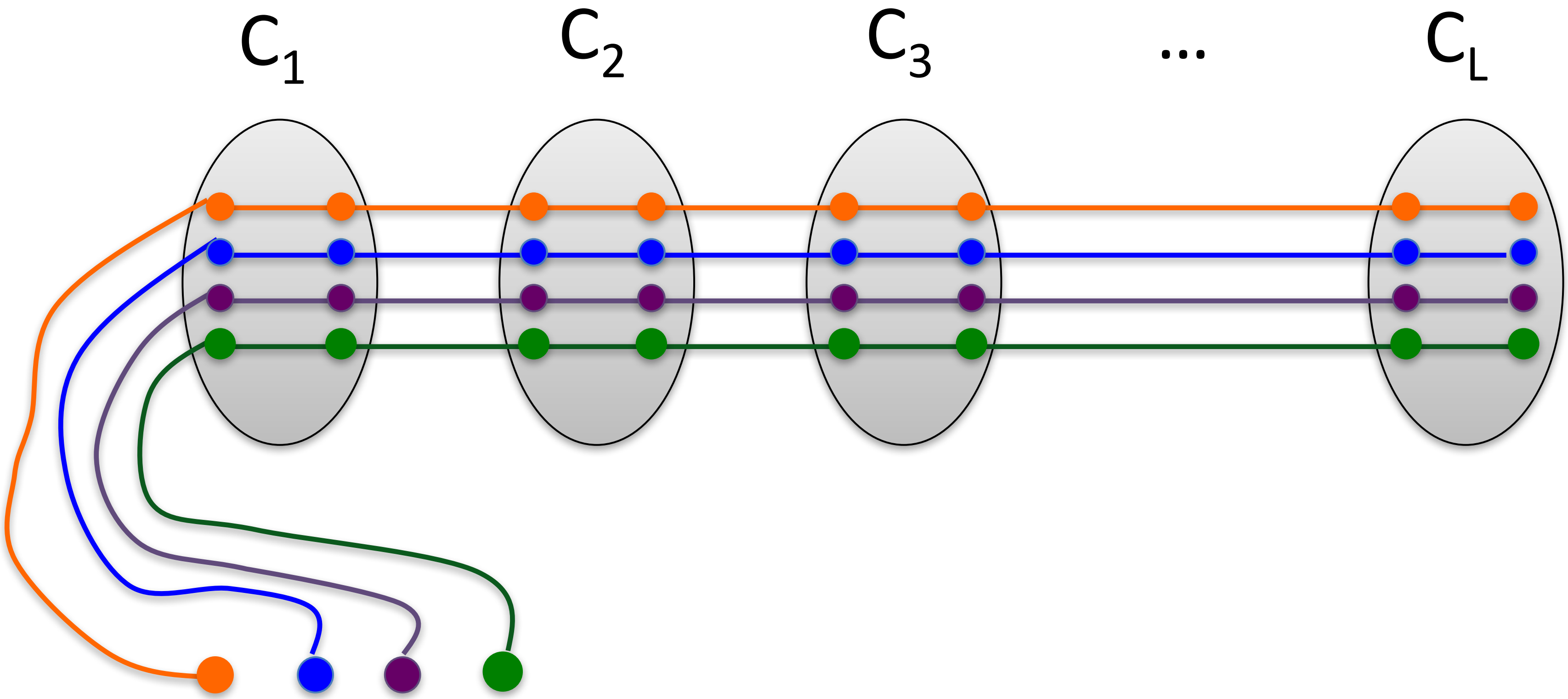
Embedding the Expander



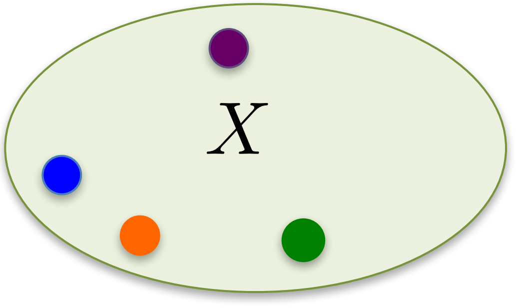
Expander vertex → the path containing the terminal



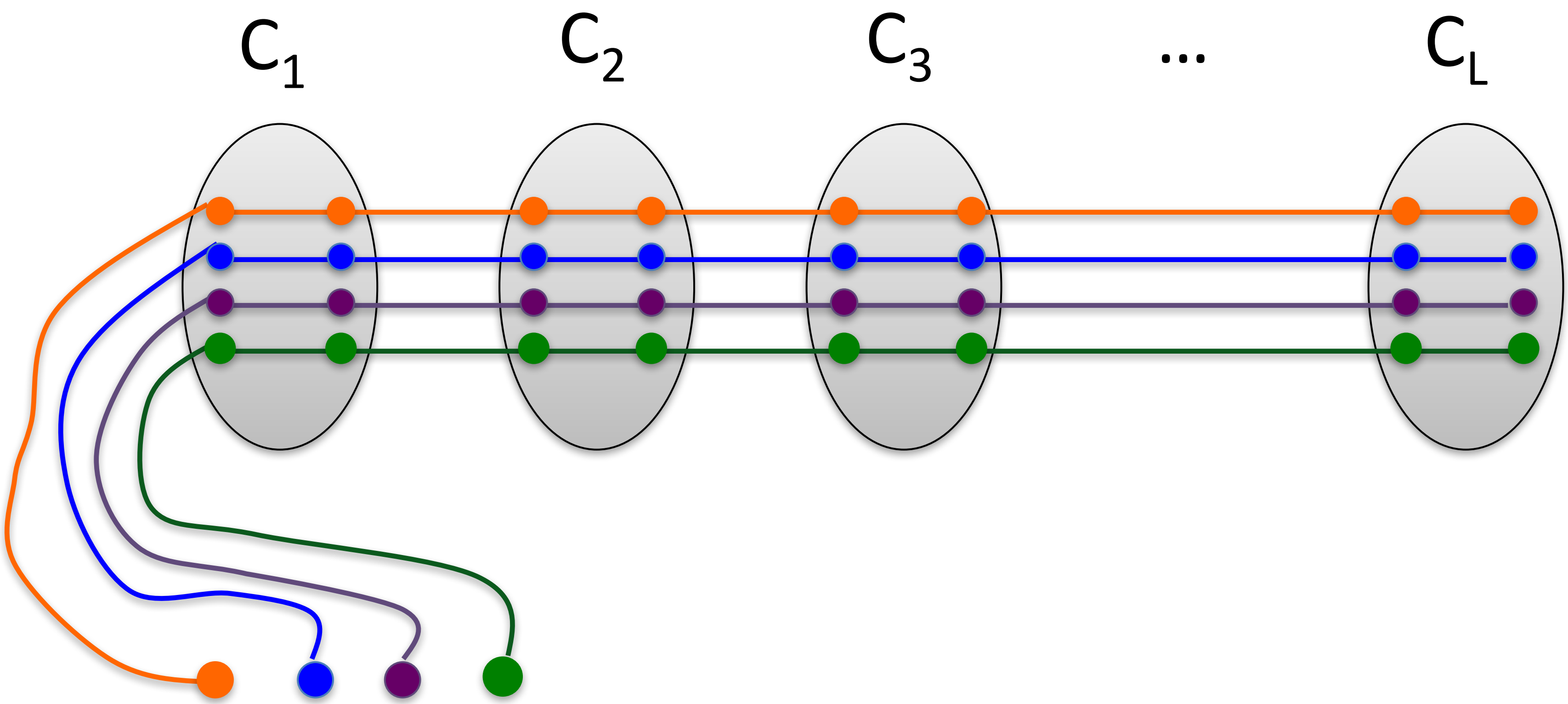
Embedding the Expander



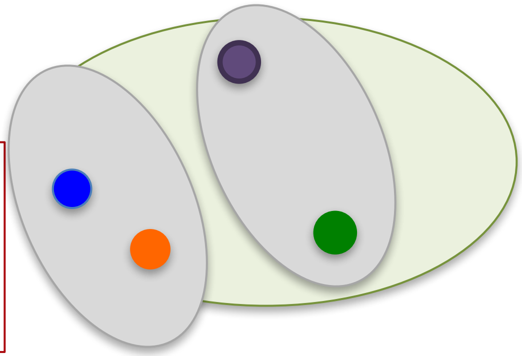
Expander vertex → the path containing the terminal



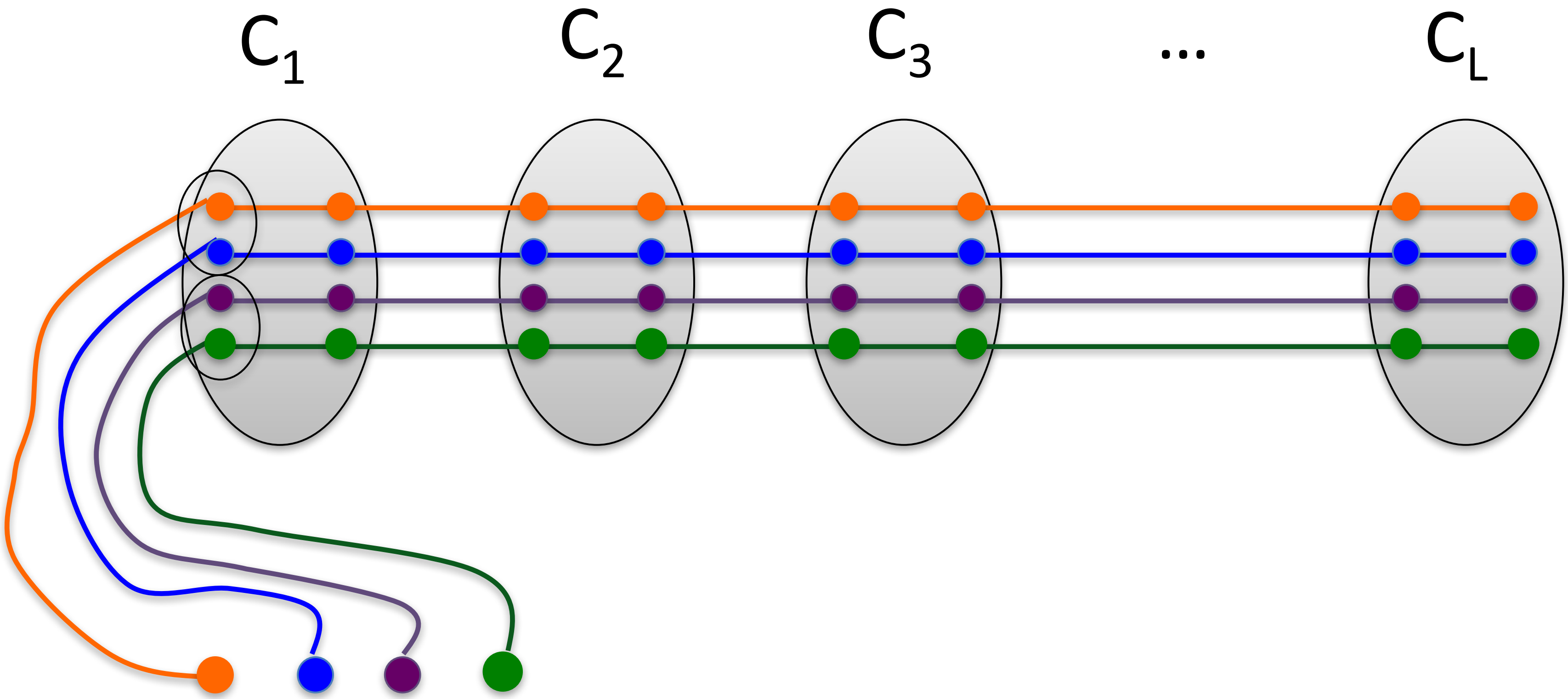
Embedding the Expander



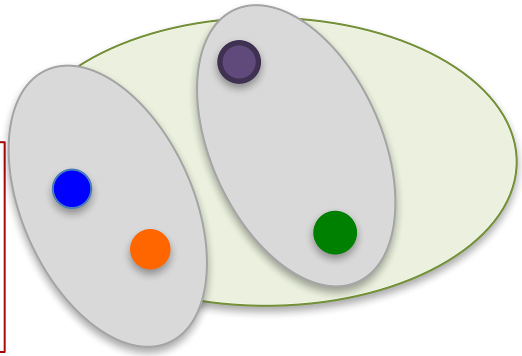
Expander edges? → cut-matching game!



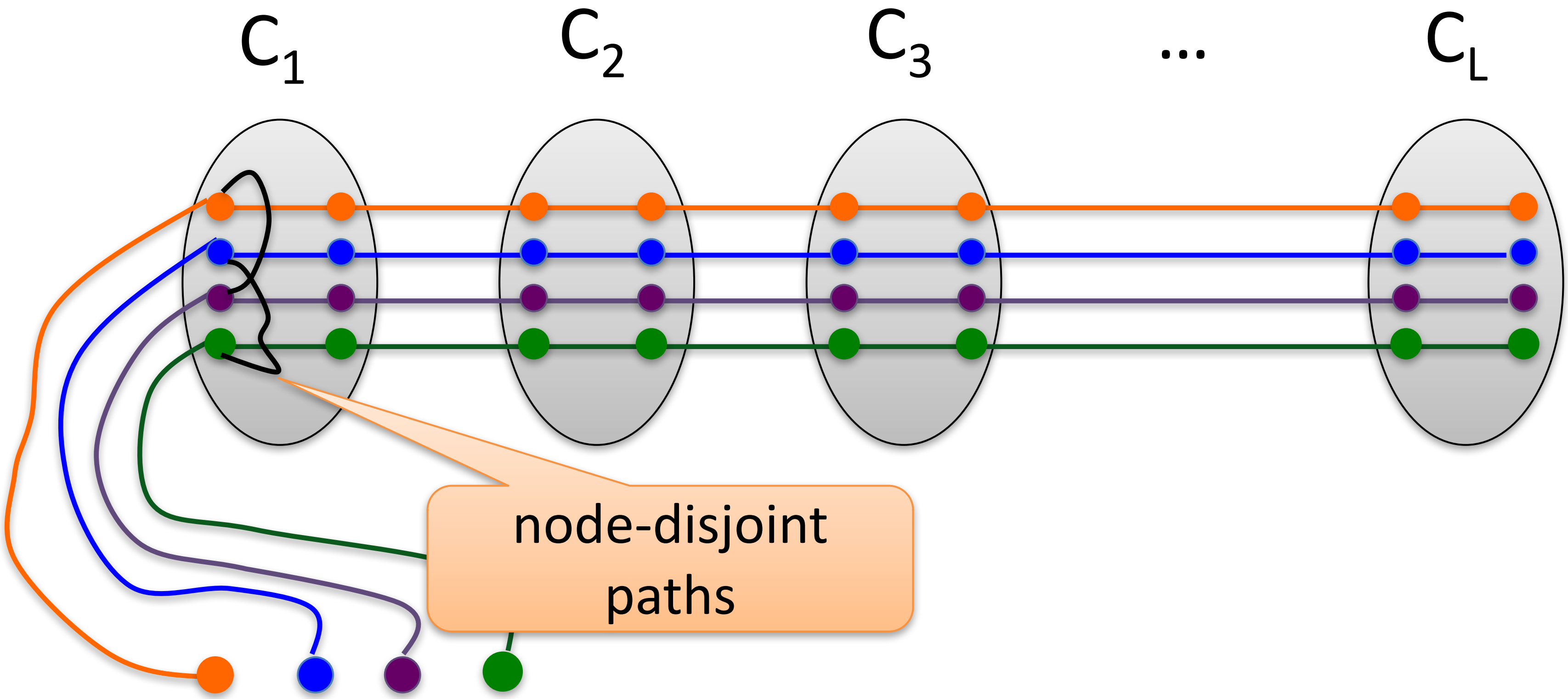
Embedding the Expander



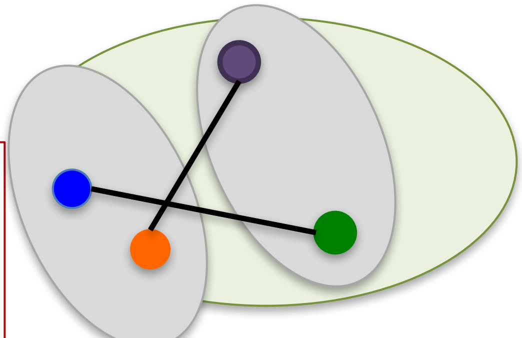
Expander edges? → cut-matching game!



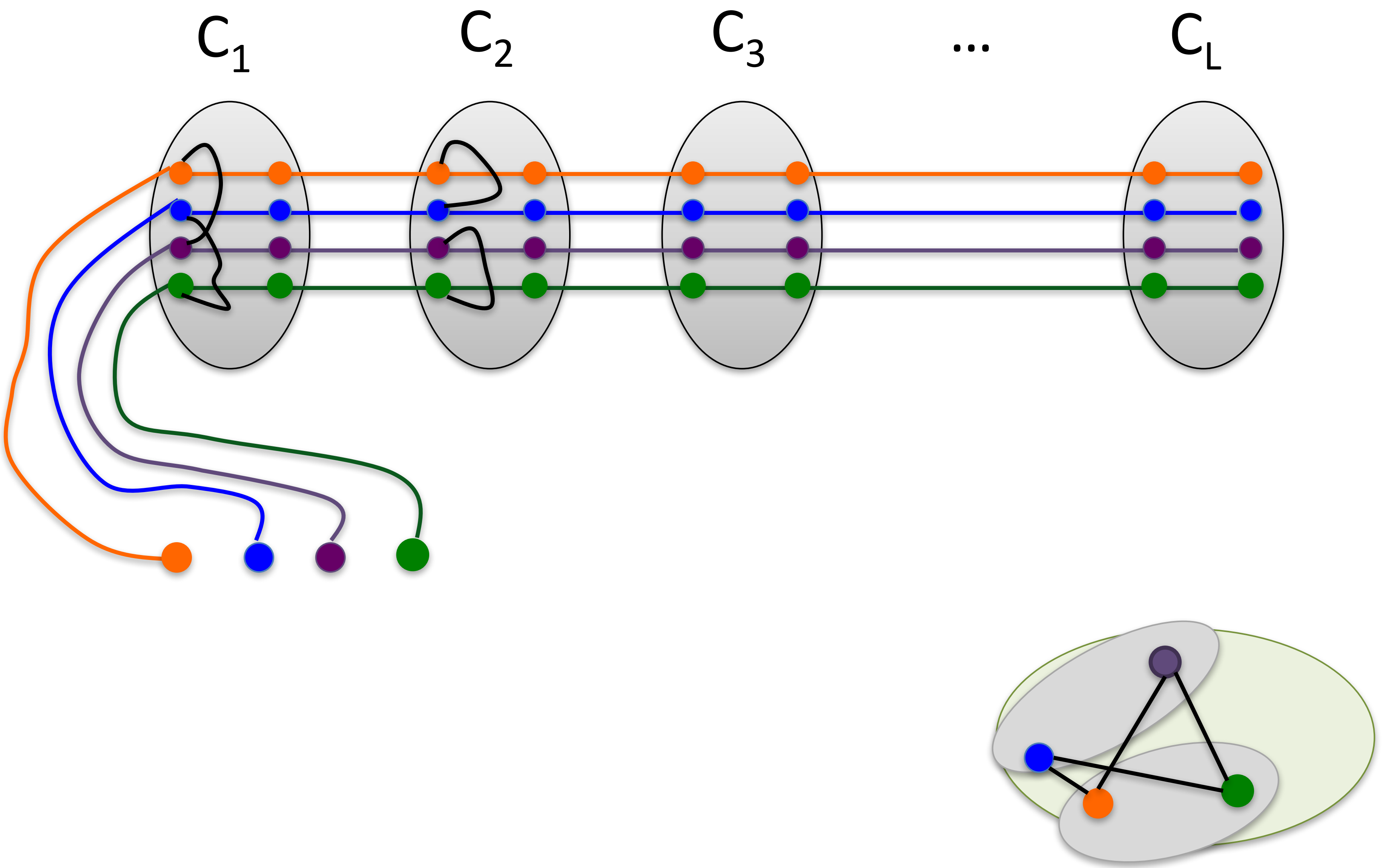
Embedding the Expander



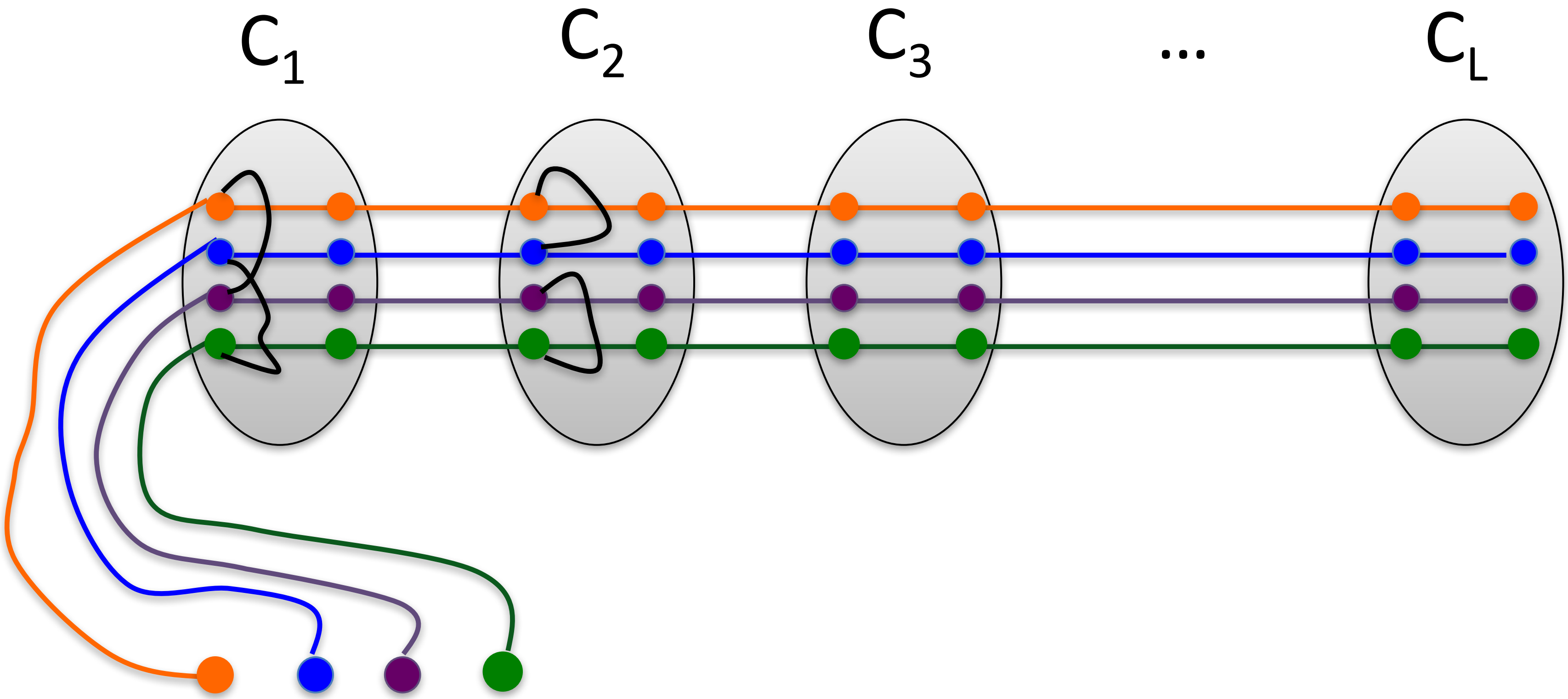
Expander edges? → cut-matching game!



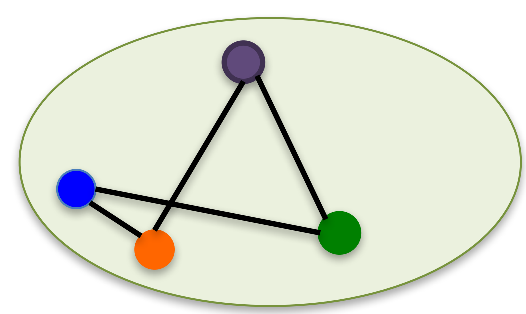
Embedding the Expander



Embedding the Expander



After $O(\log^2 k)$ iterations, we obtain an expander embedded into G with congestion 2.



Algorithm for EDPwC in Well-Linked Instances

Find a Path-of-Sets System

Embed an expander into G

Find vertex-disjoint routing in the expander

Transform into routing in G

Structural Result

If G contains a large well-linked set of vertices, then it contains a large Path-of-Sets System

Treewidth sparsifiers

Excluded grid theorem

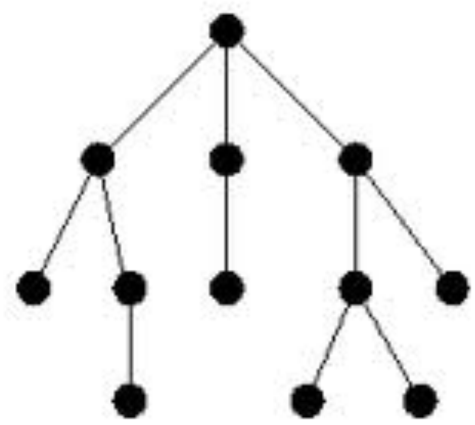
Large-treewidth graph decompositions

Vertex flow sparsifiers

Excluded Grid Theorem [Robertson, Seymour]

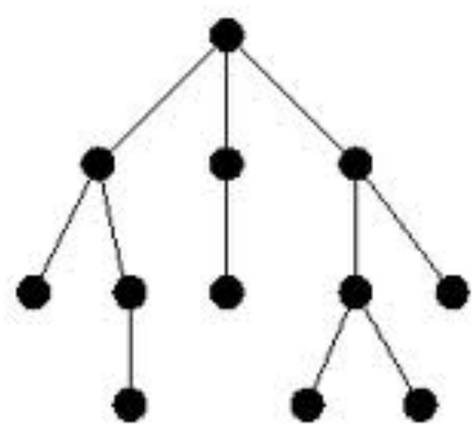
Excluded Grid Theorem [Robertson, Seymour]

Simple graphs

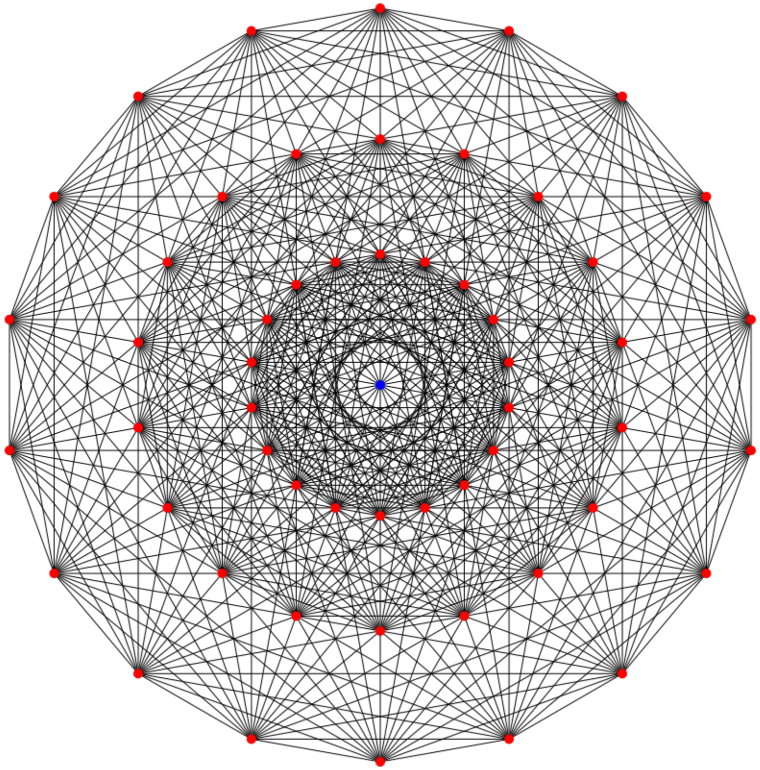


Excluded Grid Theorem [Robertson, Seymour]

Simple graphs



Complicated graphs

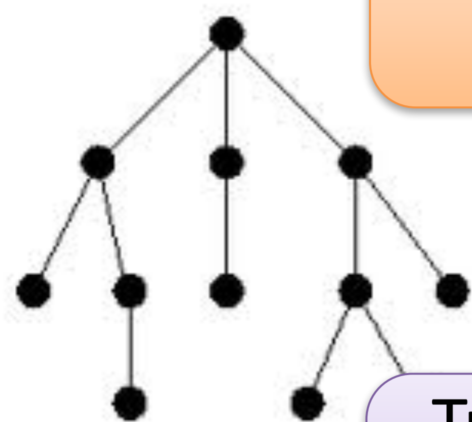


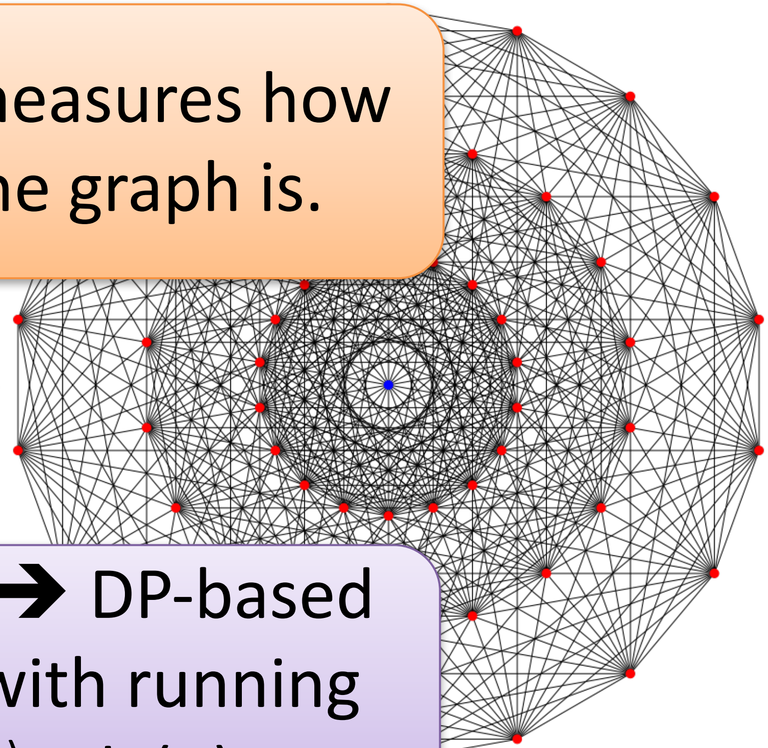
Excluded Grid Theorem [Robertson, Seymour]

Simple graphs

Complicated graphs

Treewidth: measures how complex the graph is.



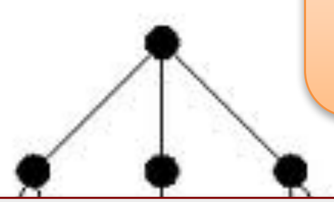


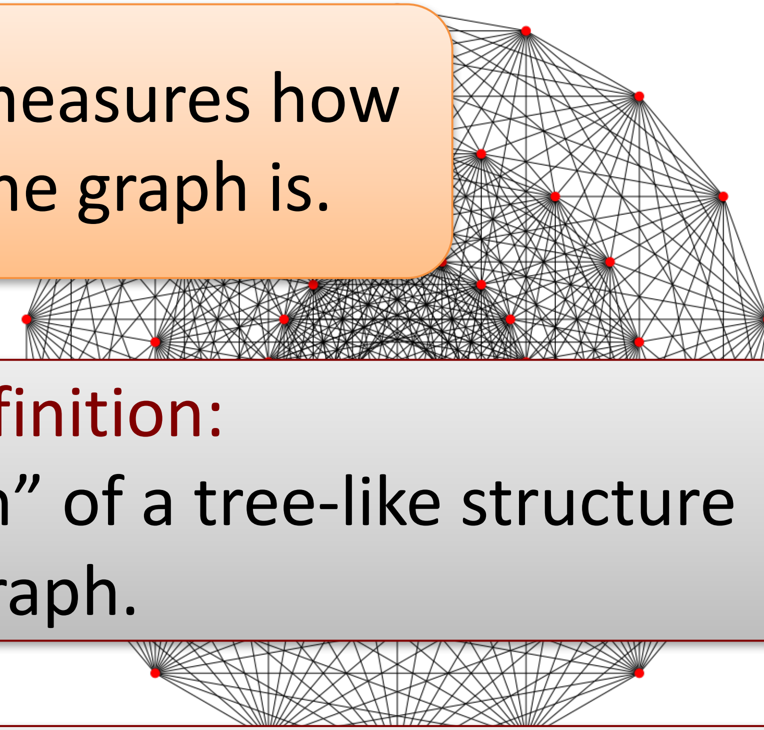
Treewidth k → DP-based algorithms with running time $2^{O(k)}$poly(n).

Excluded Grid Theorem [Robertson, Seymour]

Simple graphs

Complicated graphs





Treewidth: measures how complex the graph is.

Original definition:

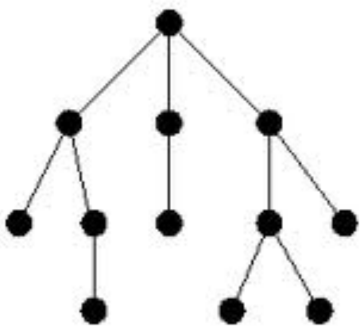
Treewidth is the smallest "width" of a tree-like structure that correctly "simulates" the graph.

(Almost) Equivalent definition:

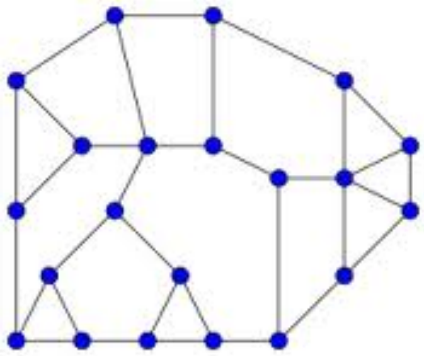
Treewidth is the cardinality of the largest well-linked set of vertices in the graph.

Treewidth

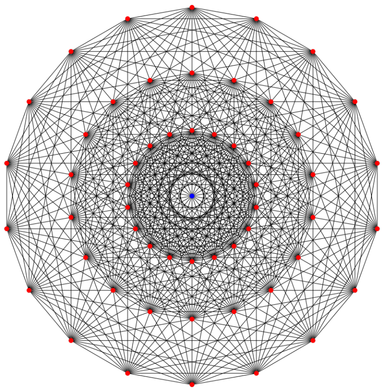
Trees



Low-Treewidth Graphs

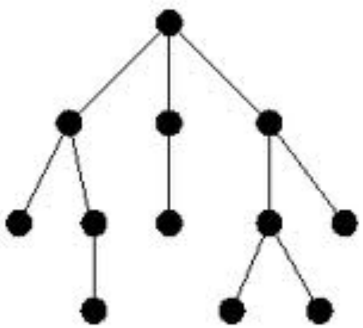


High-Treewidth Graphs

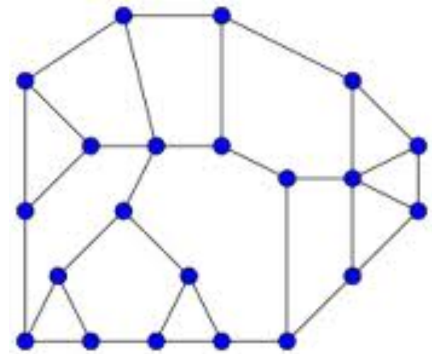


Treewidth

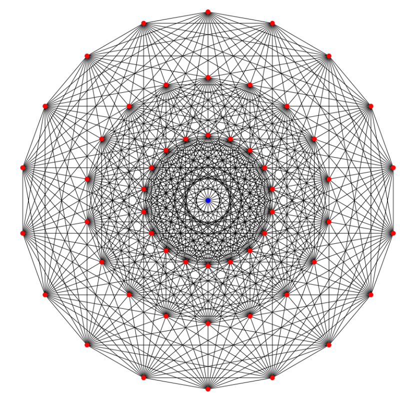
Trees



Low-Treewidth Graphs



High-Treewidth Graphs



Excluded Grid Theorem

[Robertson, Seymour '86]

If the treewidth of G is large, then G contains a large grid as a minor.

Can embed a large grid into G with no congestion

Excluded Grid Theorem [Robertson, Seymour]

If the treewidth of G is large, then it contains a large grid minor, so:

- G contains many disjoint cycles
- G contains many disjoint cycles of length 0 mod m
- G contains a convenient routing structure
- The size of the vertex cover in G is large
- …

Applications

- Fixed parameter tractability
- Erdos-Posa type results
- Graph minor theory
  - Algorithm for NDP where k is small
- Algorithms for graph crossing number
- …

Excluded Grid Theorem

[Robertson, Seymour ‘86]

If the treewidth of G is k, then G contains a grid of size f(k)xf(k) as a minor.

Excluded Grid Theorem

[Robertson, Seymour '86]

If the treewidth of G is k, then G contains a grid of size f(k)xf(k) as a minor.

How large is f(k)?

- [Robertson, Seymour '94]: $$f(k) = O\left(\sqrt{k/\log k}\right)$$
- Conjecture [Robertson, Seymour '94]: This is tight.

Excluded Grid Theorem

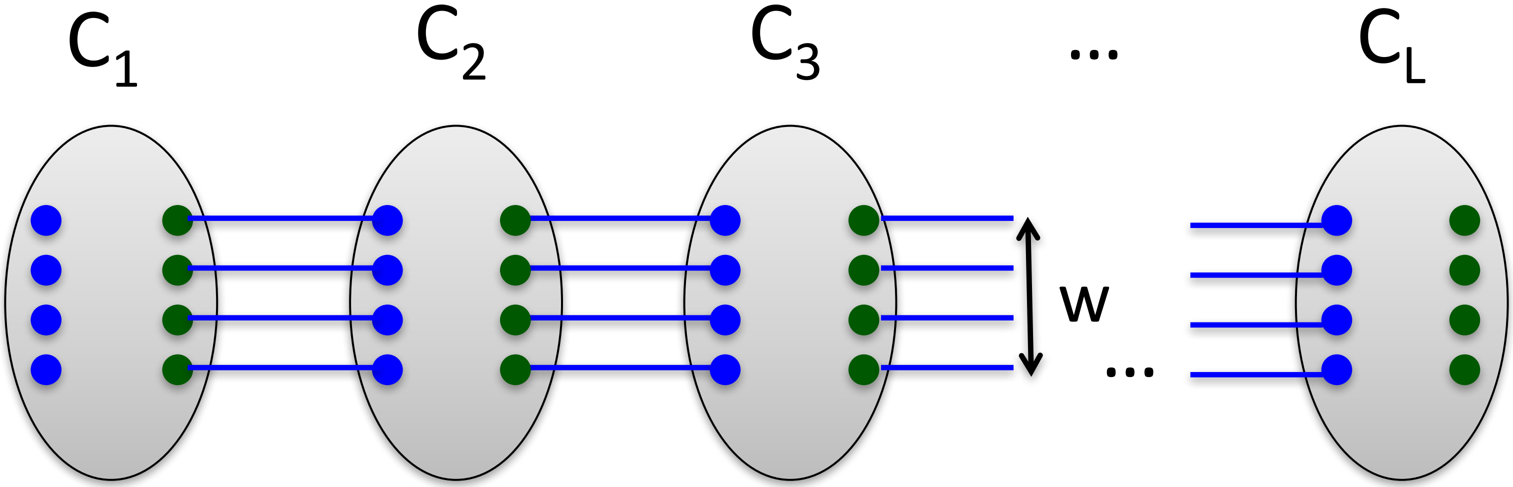
- [Robertson, Seymour, Thomas '89]: $f(k)=\Omega\left(\log^{1/5}k\right)$
- [Diestel, Gorbunov, Jensen, Thomassen '99] – simpler proof
- [Kawarabayashi, Kobayashi '12], [Leaf, Seymour '12]:

$$f(k)=\Omega\left(\sqrt{\frac{\log k}{\log\log k}}\right)$$

- [Chekuri, C '13]: $f(k)=\tilde{\Omega}\left(k^{1/98}\right)$
- [C, '16]: $f(k)=\tilde{\Omega}\left(k^{1/19}\right)$

Main Idea

width w
length L

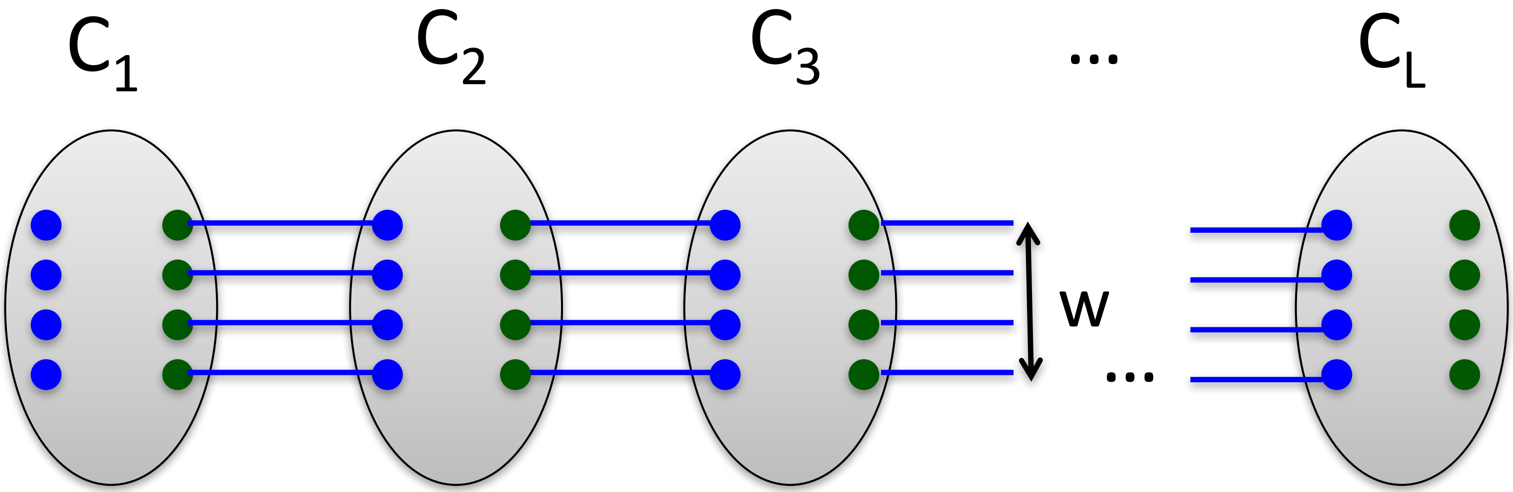


Thm: If G contains a path-of-sets system of width and length $\Theta(g^2)$, then there is a (gxg)-grid minor in G.

[Leaf, Seymour '12]
[Chekuri, C '13]

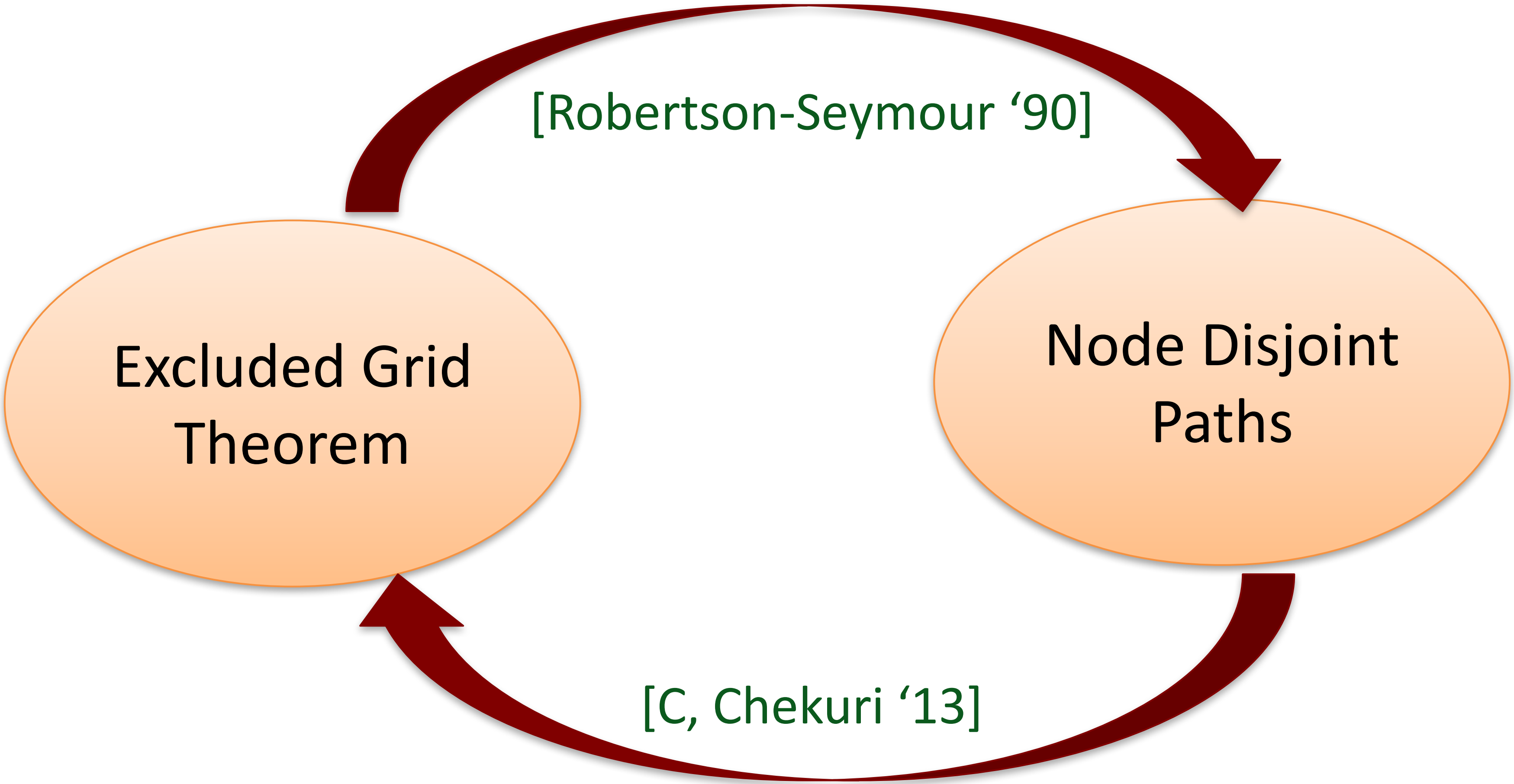
Main Idea

width w
length L



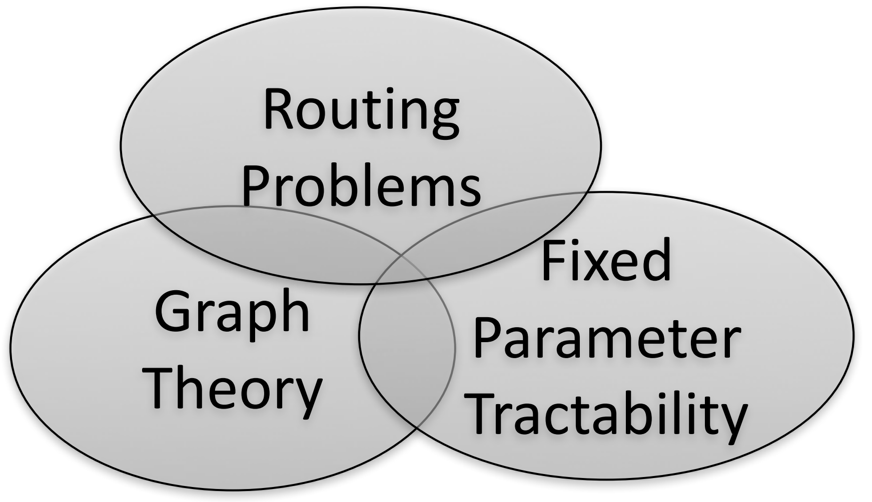
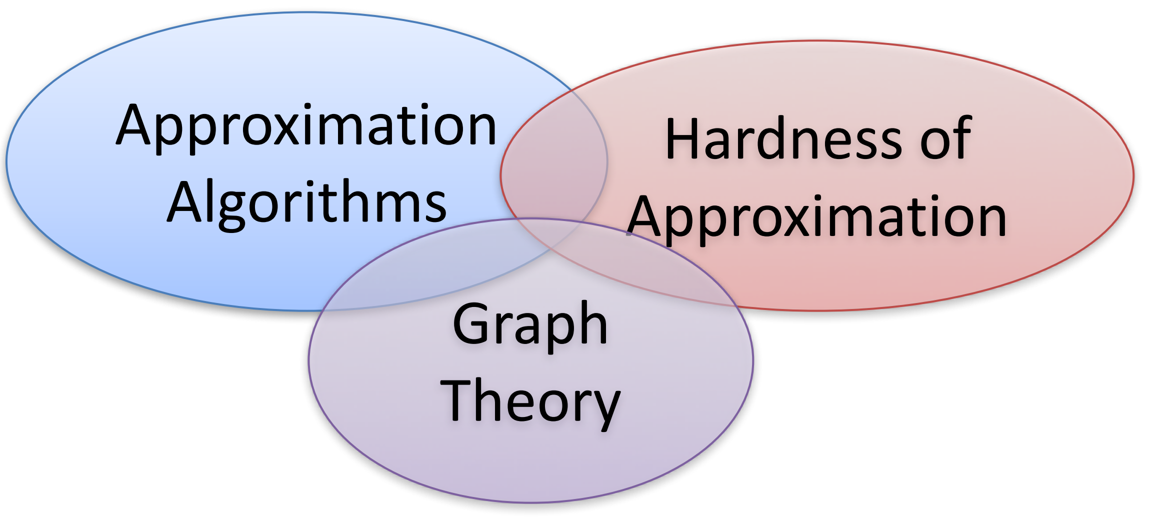
Thm: If G contains a path-of-sets system of width and length $\Theta(g^2)$, then there is a (gxg)-grid minor in G.

G has large treewidth → large well-linked set of vertices → large Path-of-Sets system


[Robertson-Seymour ‘90]
Excluded Grid Theorem
Node Disjoint Paths
[C, Chekuri ‘13]

Historical Note

- Work on routing gave slightly weaker structure than Path-of-Sets System, called Tree-of-Sets System
- We later modified it to get the Path-of-Sets system for the Excluded Grid theorem.
- This in turn helped improve results for routing problems.


Routing Problems
Graph Theory
Fixed Parameter Tractability

Approximation Algorithms
Hardness of Approximation
Graph Theory

Open Problems

- Getting tight bounds for the Excluded Grid Theorem.
- Simpler algorithms for NDP with constant k
- Congestion minimization:
  - $O(\log n / \log\log n)$-approximation algorithm
  - $\Omega(\log\log n)$-hardness of approximation
  - Integrality gap of the multicommodity LP relaxation open

Open Problems

- Getting tight bounds for the Excluded Grid Theorem.
- Simpler algorithms for NDP wit k
- Congestion minimization:
  - O(log n/log log n)-approximation algorithm
  - Ω(log log n)-hardness of approximation
  - Integrality gap of the multicommodity LP relaxation open

Planar graphs?

Thank you!