The Subtour LP for the Traveling Salesman Problem

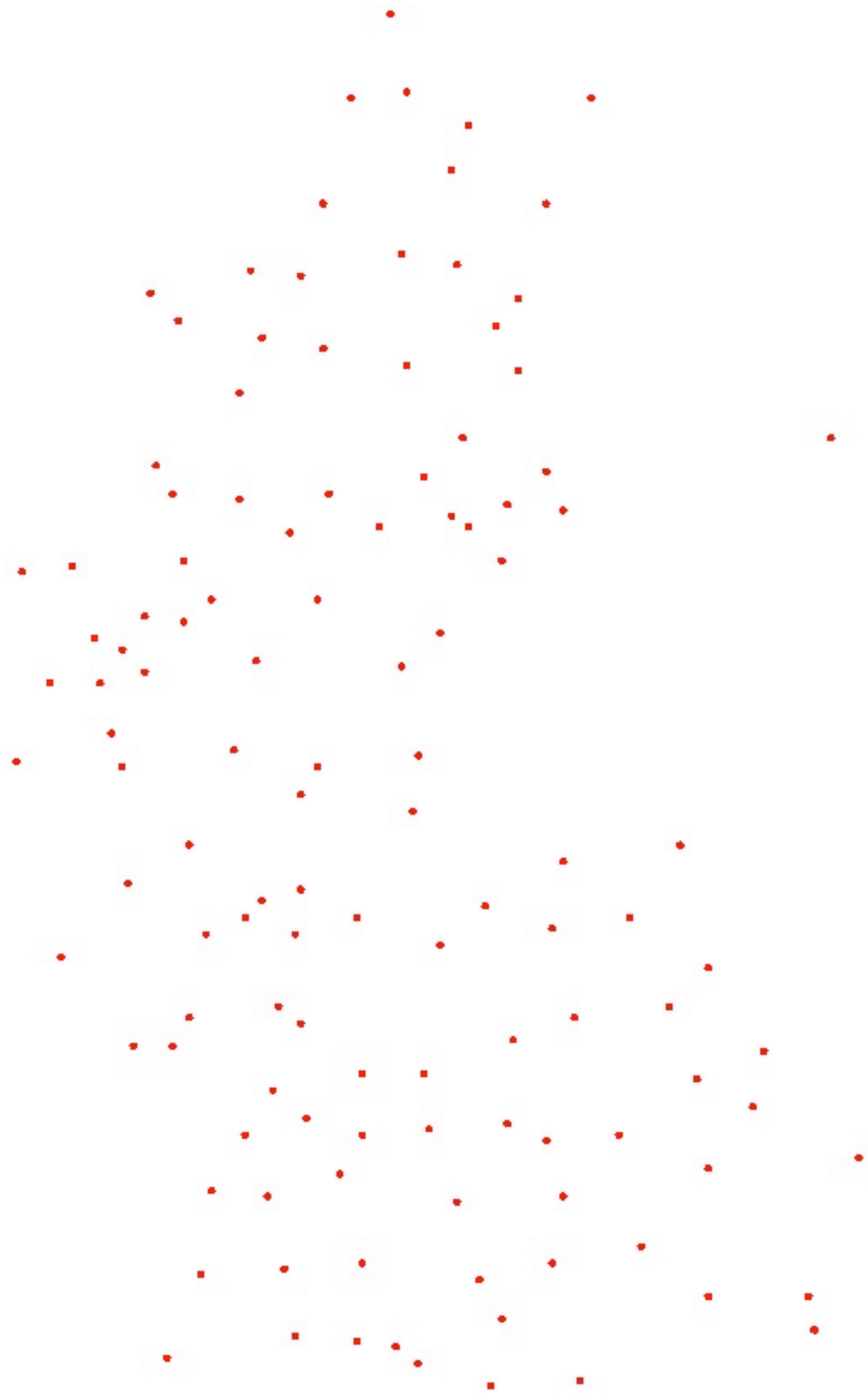
David P. Williamson
Cornell University
November 22, 2011

Joint work with Jiawei Qian, Frans Schalekamp, and Anke van Zuylen

The Traveling Salesman Problem

The most famous problem in discrete optimization: Given *n* cities and the cost *c(i,j)* of traveling from city *i* to city *j*, find a minimum-cost tour that visits each city exactly once.

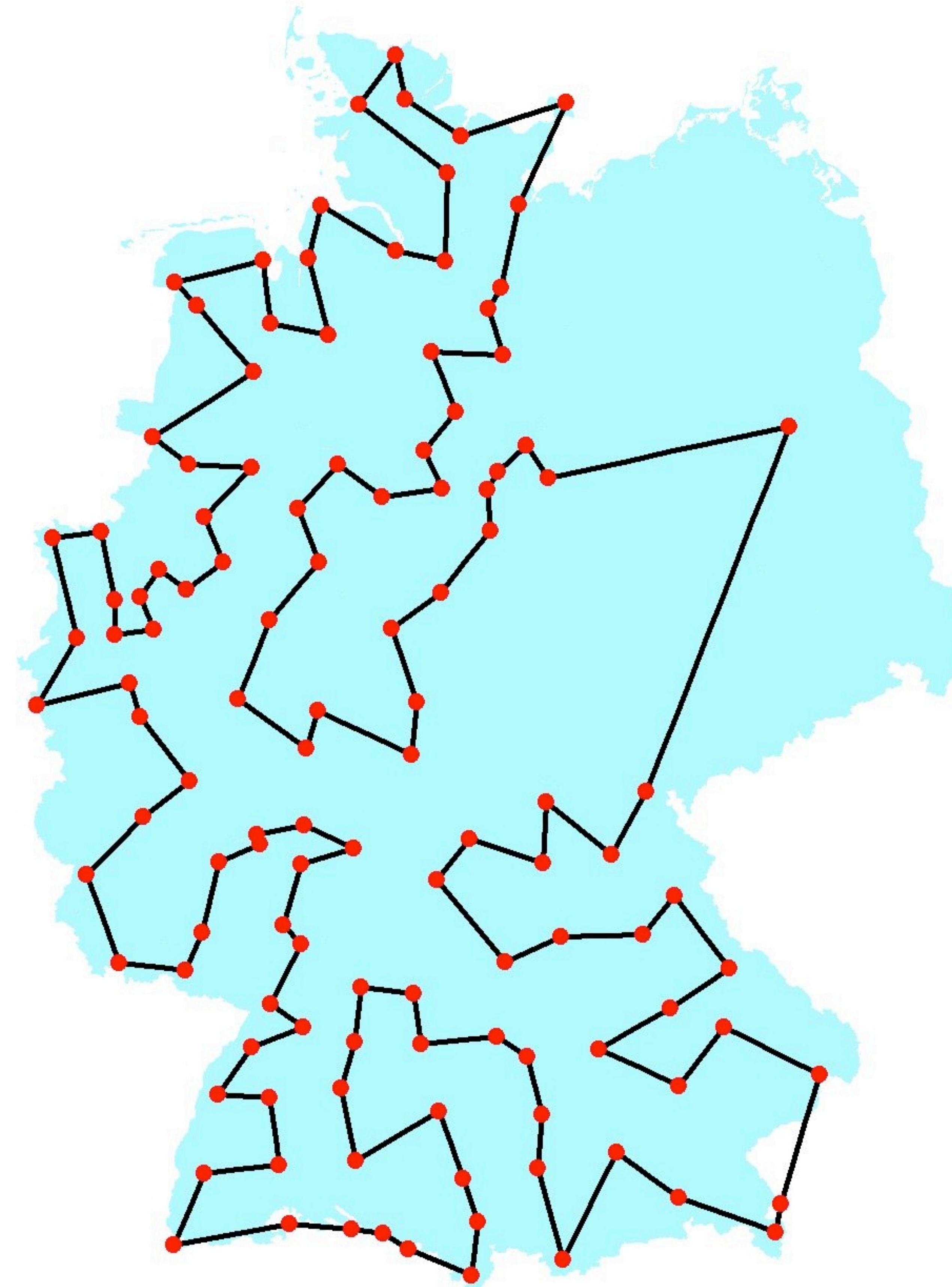
We assume costs are symmetric (*c(i,j)=c(j,i)* for all i,j) and obey the triangle inequality ($c(i,j) \leq c(i,k) + c(k,j)$ for all i,j,k).



The Traveling Salesman Problem

The most famous problem in discrete optimization: Given *n* cities and the cost *c(i,j)* of traveling from city *i* to city *j*, find a minimum-cost tour that visits each city exactly once.

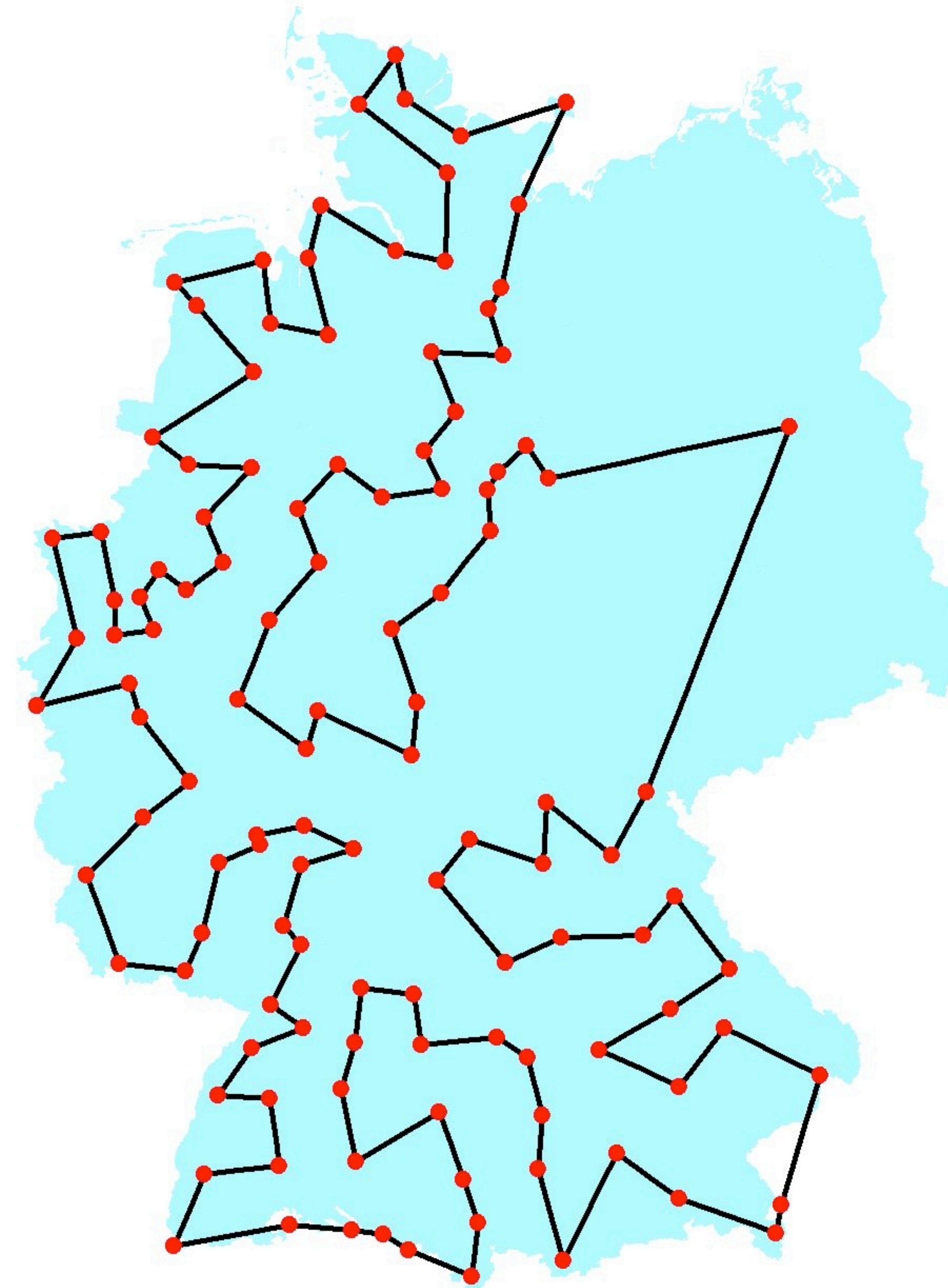
We assume costs are symmetric (*c(i,j)*=*c(j,i)* for all i,j) and obey the triangle inequality (*c(i,j)* ≤ *c(i,k)* + *c(k,j)* for all i,j,k).



The Traveling Salesman Problem

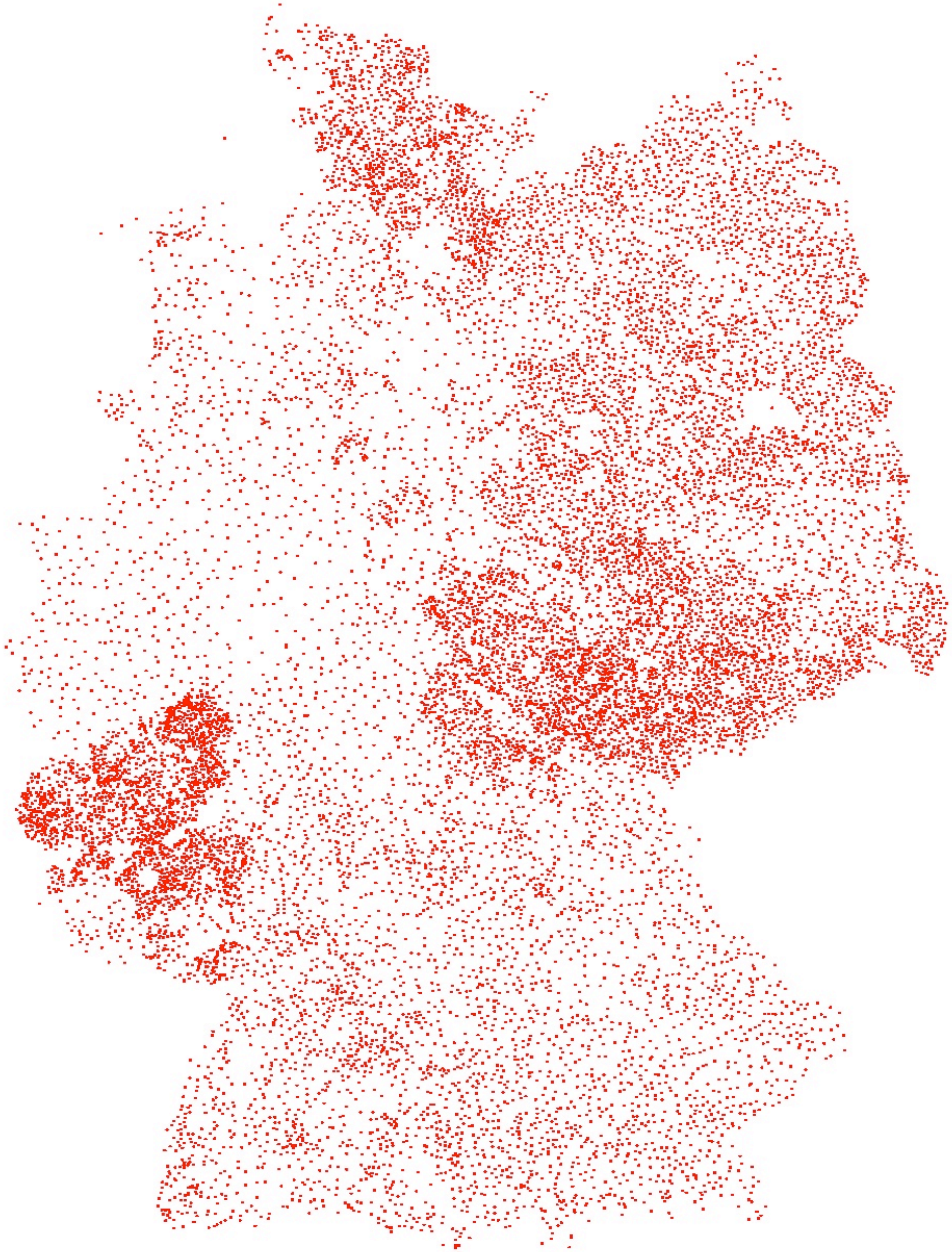
The most famous problem in discrete optimization: Given *n* cities and the cost *c(i,j)* of traveling from city *i* to city *j*, find a minimum-cost tour that visits each city exactly once.

We assume costs are symmetric ($c(i,j)=c(j,i)$ for all i,j) and obey the triangle inequality ($c(i,j) \leq c(i,k) + c(k,j)$ for all i,j,k).

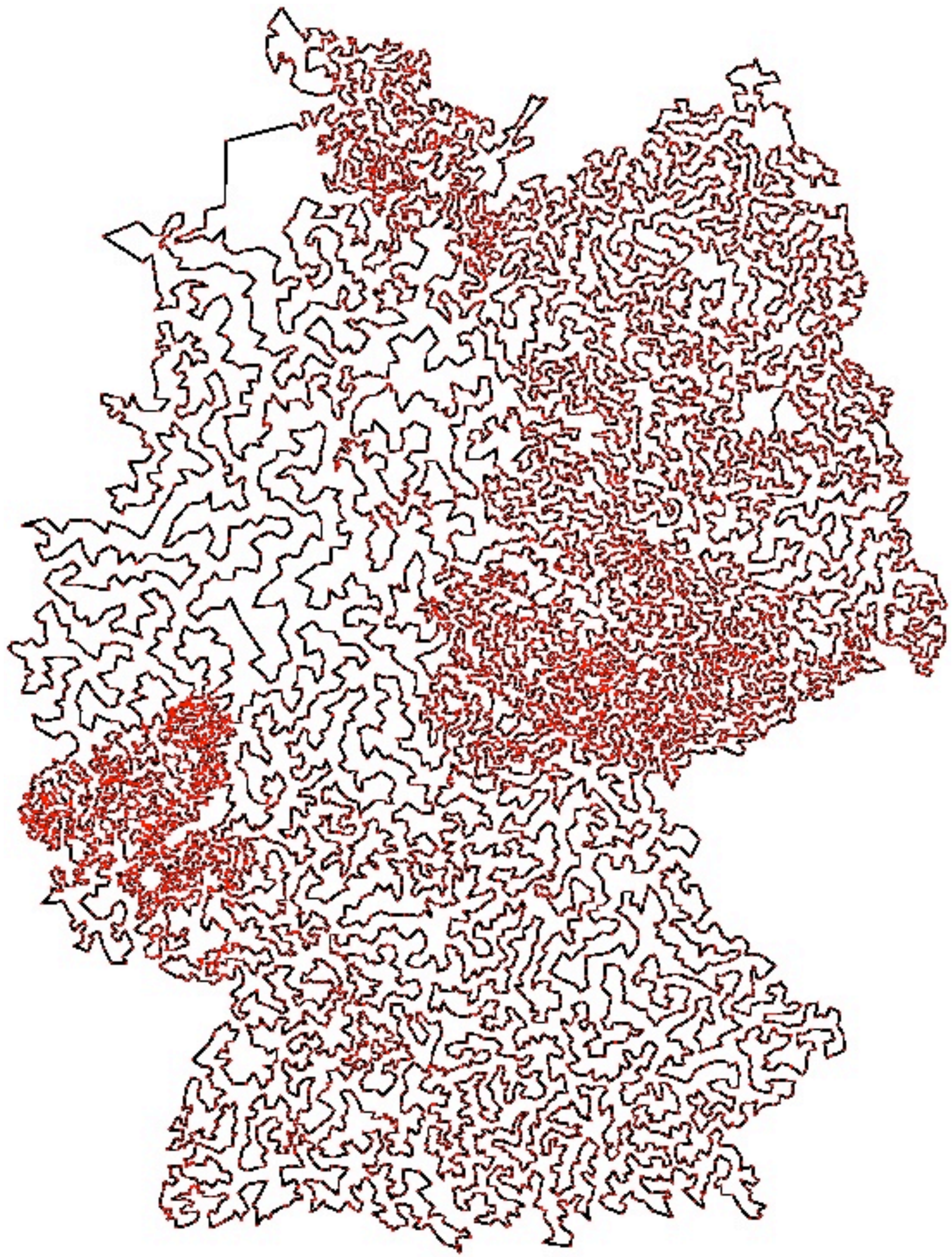


120 city tour of West Germany due to M. Grötschel (1977)

A 15112 city instance solved by Applegate, Bixby, Chvátal, and Cook (2001)

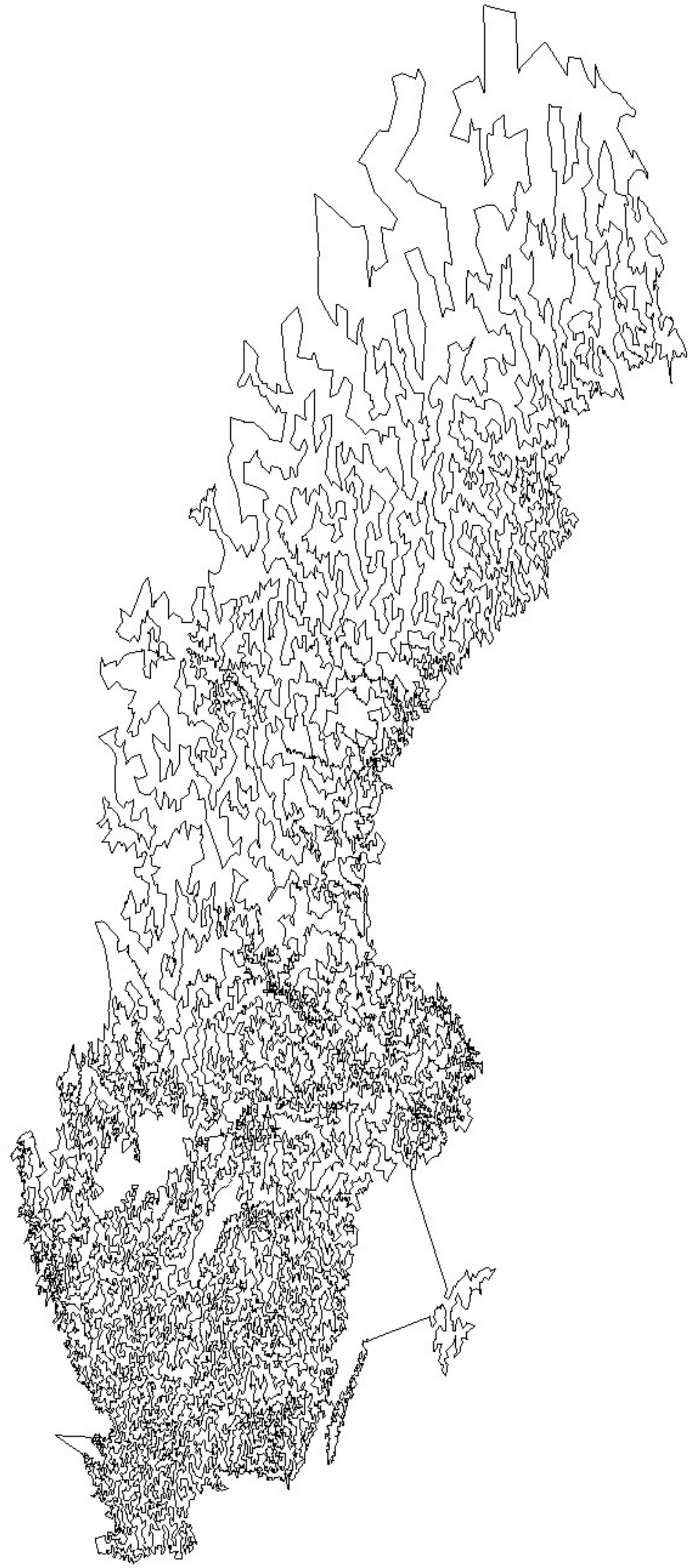


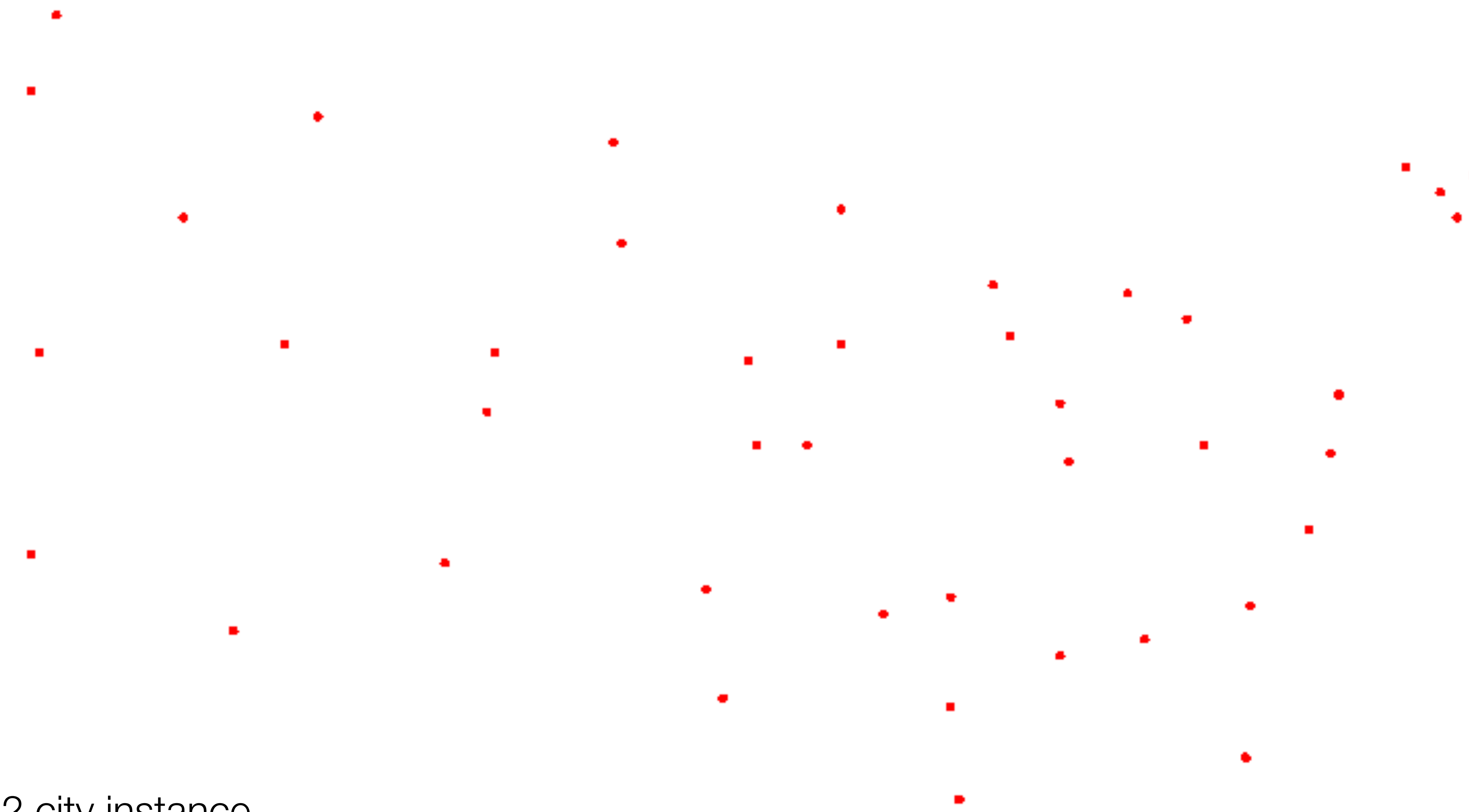
A 15112 city instance solved by Applegate, Bixby, Chvátal, and Cook (2001)



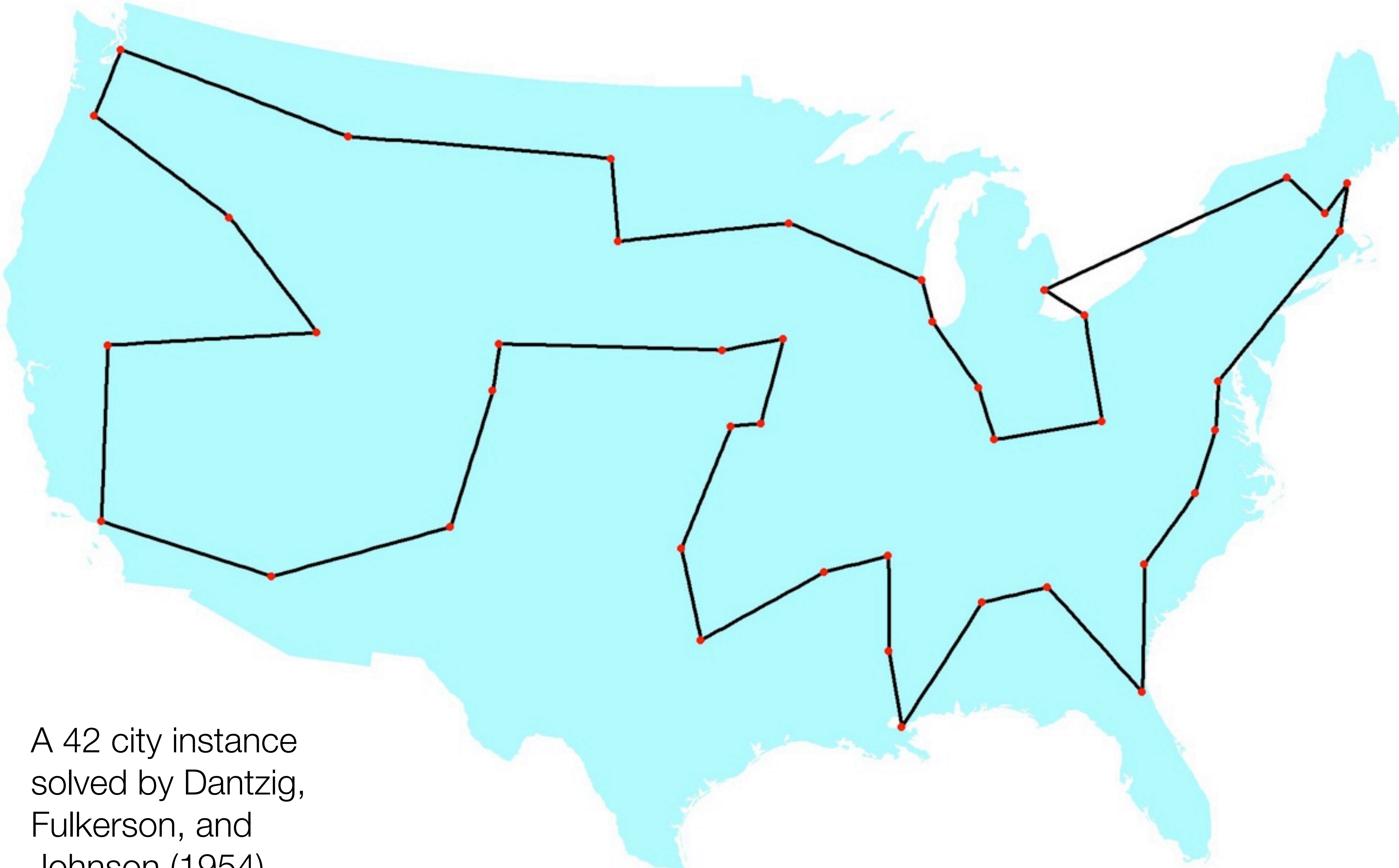
A 24978 city instance from Sweden solved by Applegate, Bixby, Chvátal, Cook, and Helsgaun (2004)

A 24978 city instance from Sweden solved by Applegate, Bixby, Chvátal, Cook, and Helsgaun (2004)





A 42 city instance
solved by Dantzig,
Fulkerson, and
Johnson (1954)



A 42 city instance solved by Dantzig, Fulkerson, and Johnson (1954)

The Dantzig-Fulkerson-Johnson Method

- $G=(V,E)$ is a complete graph on *n* vertices
- $c(e)=c(i,j)$ is the cost of traveling on edge $e=(i,j)$
- $x(e)$ is a decision variable indicating if edge e is used in the tour, $0 \leq x(e) \leq 1$
- Solve linear program; if $x(e)$ are integer tour, stop, else find a *cutting plane*

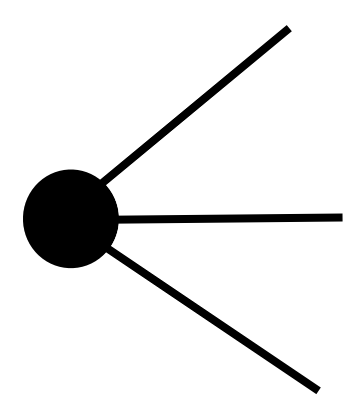
The linear program

$$\text{Minimize} \sum_{e \in E} c(e)x(e)$$

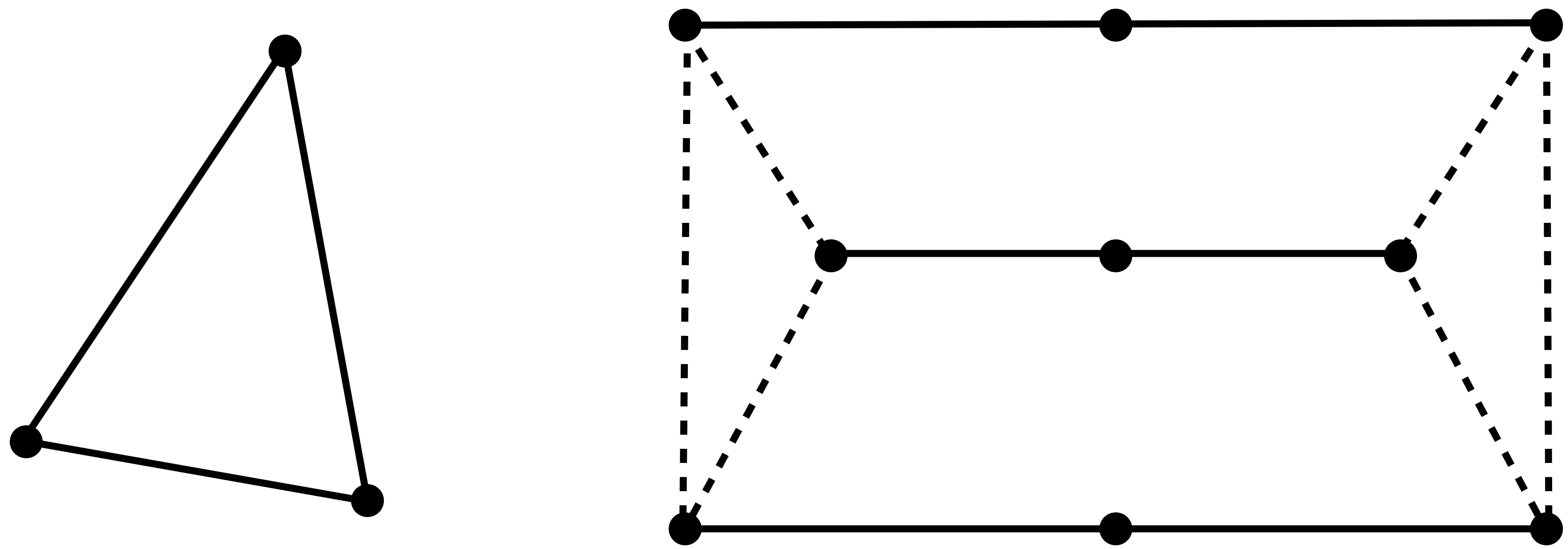
subject to

$$\sum_{e \in \delta(v)} x(e) = 2 \qquad \forall v \in V$$

$$0 \leq x(e) \leq 1 \qquad \forall e \in E$$

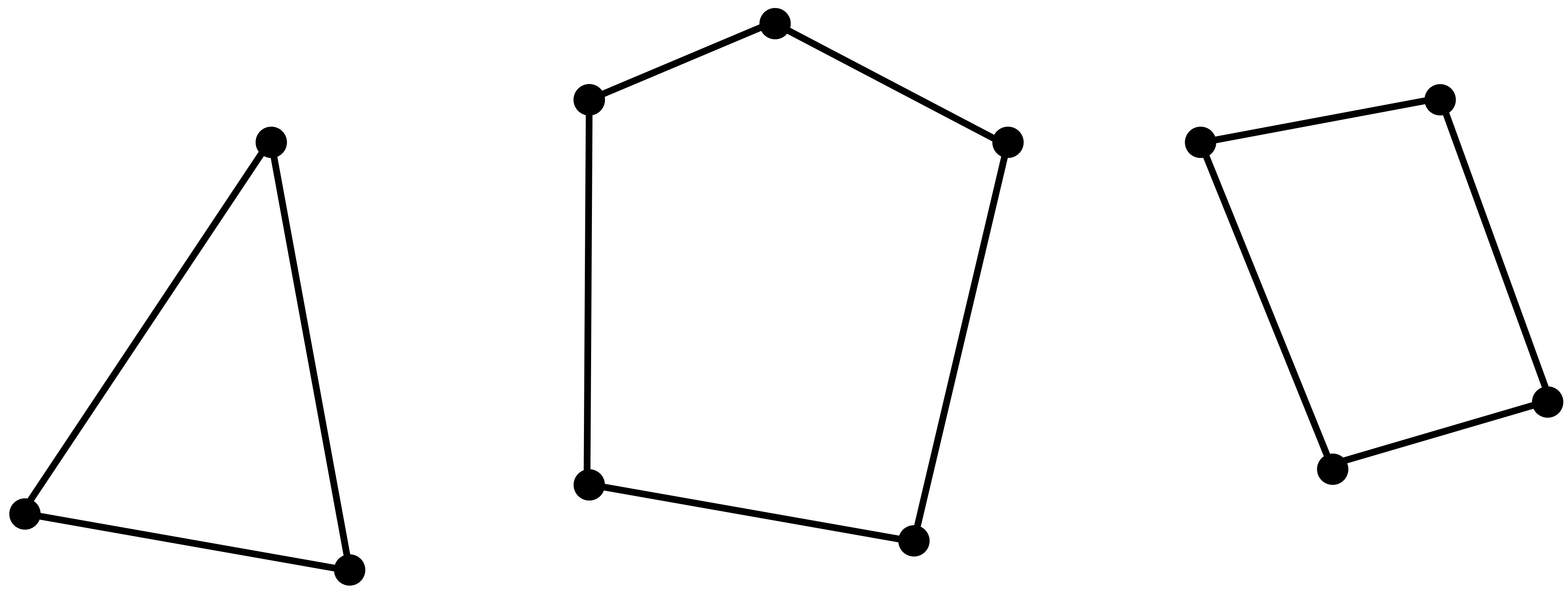


Fractional 2-matchings



Fractional (basic) solutions have components that are cycles of size at least 3 with x(e)=1 or odd cycles with x(e)=1/2 connected by paths with x(e)=1

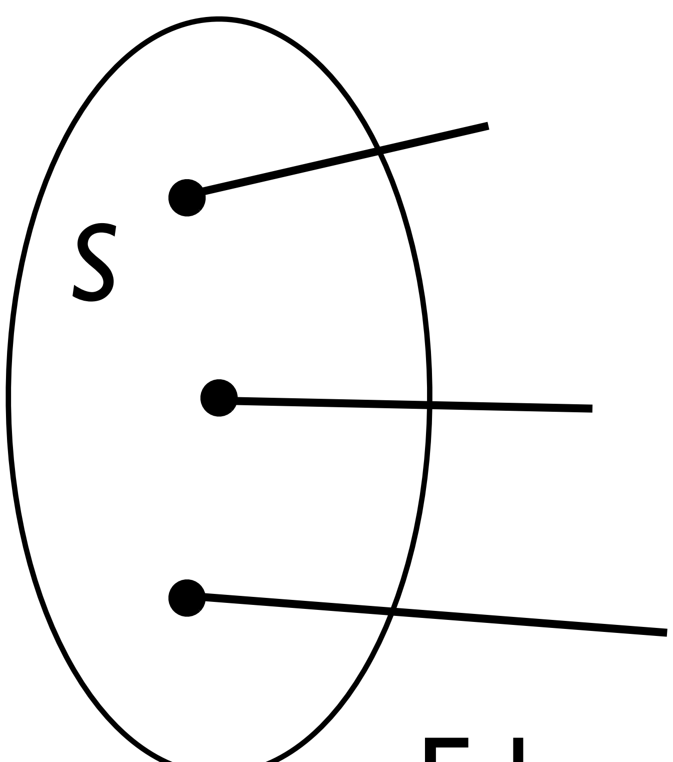
2-matchings



Integer solutions have components with cycles of size at least 3; sometimes called *subtours*

“Loop conditions”

Dantzig, Fulkerson, and Johnson added constraints to eliminate subtours as they occurred; these now called “subtour elimination constraints”.



$$\sum_{e \in \delta(S)} x(e) \geq 2 \qquad \forall S \subseteq V, |S| \geq 2$$

Edges in the *cut* for *S*

Subtour LP

$$\text{Minimize } \sum_{e \in E} c(e)x(e)$$

subject to

$$\sum_{e \in \delta(v)} x(e) = 2 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} x(e) \geq 2 \qquad \forall S \subseteq V, |S| \geq 2$$

$$0 \leq x(e) \leq 1 \qquad \forall e \in E$$

How strong is the Subtour LP bound?

Johnson, McGeoch, and Rothberg (1996) and Johnson and McGeoch (2002) report experimentally that the Subtour LP is very close to the optimal.

Random Uniform Euclidean				TSPLIB			
Name	%Gap	Opttime	HKtime	Name	%Gap	Opttime	HKtime
E1k.0	0.77	1406	2.13	dsj1000	0.61	410	3.68
E1k.1	0.64	3855	2.15	pr1002	0.89	34	2.40
E1k.2	0.72	1211	2.02	si1032	0.08	25	11.32
E1k.3	0.62	956	1.92	u1060	0.65	571	3.62
E1k.4	0.69	330	1.69	vm1084	1.33	605	2.40
E1k.5	0.59	233	2.42	pcb1173	0.96	468	1.70
E1k.6	0.79	2940	1.67	d1291	1.18	27394	4.54
E1k.7	0.94	8003	1.95	rl1304	1.55	189	4.08
E1k.8	1.01	4347	1.65	rl1323	1.65	3742	4.49
E1k.9	0.61	189	2.14	nrw1379	0.43	578	2.40
E3k.0	0.71	533368	9.57	fl1400	1.74	1549	9.83
E3k.1	0.67	425631	10.54	u1432	0.29	224	2.42
E3k.2	0.74	342370	9.41	fl1577	1.66	6705	38.19
E3k.3	0.67	147135	10.30	d1655	0.94	263	6.51
E3k.4	0.73		8.07	vm1748	1.35	2224	4.43
Random Clustered Euclidean				u1817	0.90	449231	5.01
C1k.0	0.54	337	9.83	rl1889	1.55	10023	11.45
C1k.1	0.41	534	10.84	d2103	1.44	–	8.19
C1k.2	0.42	320	8.79	u2152	0.62	45205	8.10
C1k.3	0.53	214	7.63	u2319	0.02	7068	3.16
C1k.4	0.58	768	9.36	pr2392	1.22	117	5.75
C1k.5	0.58	139	9.29	pcb3038	0.81	80829	7.26
C1k.6	0.73	1247	7.07	fl3795	1.04	69886	123.66
C1k.7	0.58	449	13.24	fnl4461	0.55	–	12.47
C1k.8	0.34	140	10.40	rl5915	1.56	–	42.00
C1k.9	0.66	703	9.61	rl5934	1.38		56.15
C3k.0	0.62	16009	53.03	pla7397	0.58	–	55.42
C3k.1	0.61	17754	126.49	rl11849	1.02		102.41
C3k.2	0.70	18237	80.39	usa13509	0.66	–	120.20
C3k.3	0.57	6349	71.57	d15112	0.52		90.13
C3k.4	0.57	4845	44.02				
Random Matrices							
M1k.0	0.01	60	5.47	M3k.0	0.00	612	40.35
M1k.1	0.03	137	5.51	M3k.1	0.01	546	39.52
M1k.2	0.01	151	5.63	M10k.0	0.00	1377	367.84
M1k.3	0.01	169	5.26				

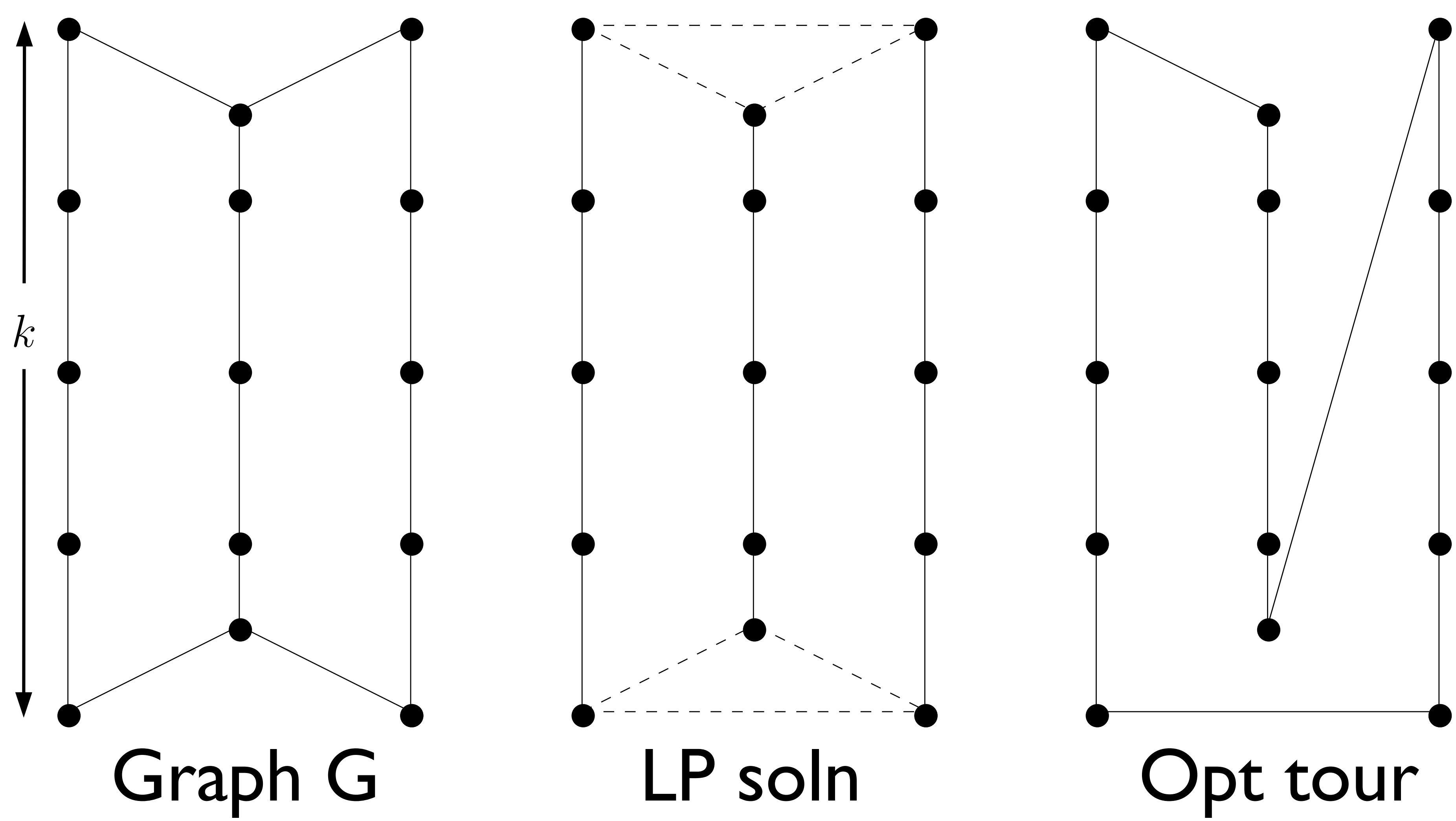
How strong is the Subtour LP bound?

- What about in theory?
- Define
  - SUBT(c) as the optimal value of the Subtour LP for costs c
  - OPT(c) as the length of the optimal tour for costs c
  - $\mathcal{C}_n$ is the set of all symmetric cost functions on *n* vertices that obey triangle inequality.
- Then the *integrality gap* of the Subtour LP is

$$\gamma \equiv \sup_n \gamma(n) \text{ where } \gamma(n) \equiv \sup_{c \in \mathcal{C}_n} \frac{OPT(c)}{SUBT(c)}$$

A lower bound

It's known that $\gamma \geq 4/3$, where *c(i,j)* comes from the shortest *i-j* path distance in a graph G (*graphic TSP*).

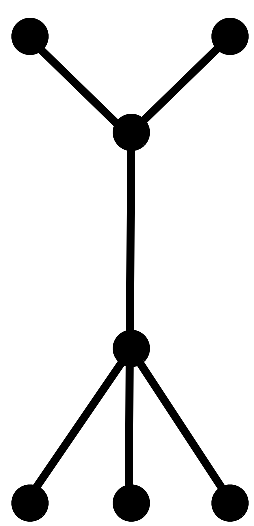


Christofides' Algorithm

Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.

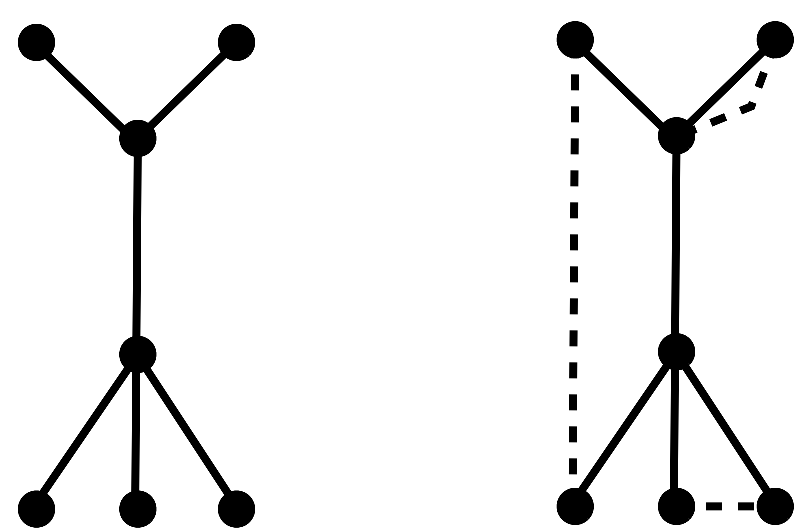
Christofides' Algorithm

Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



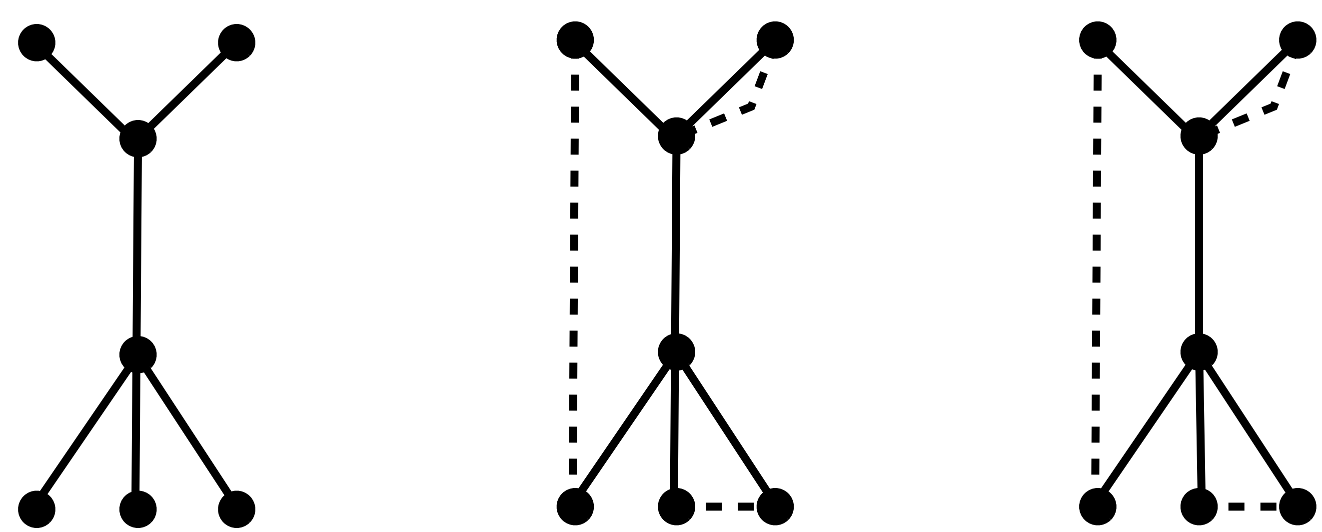
Christofides' Algorithm

Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



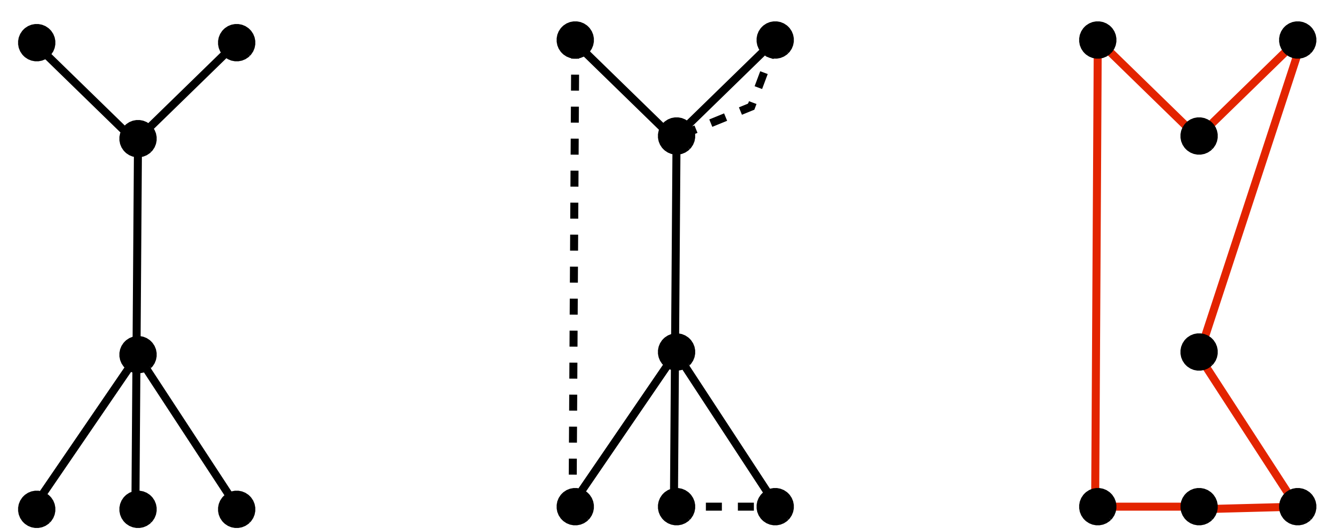
Christofides' Algorithm

Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



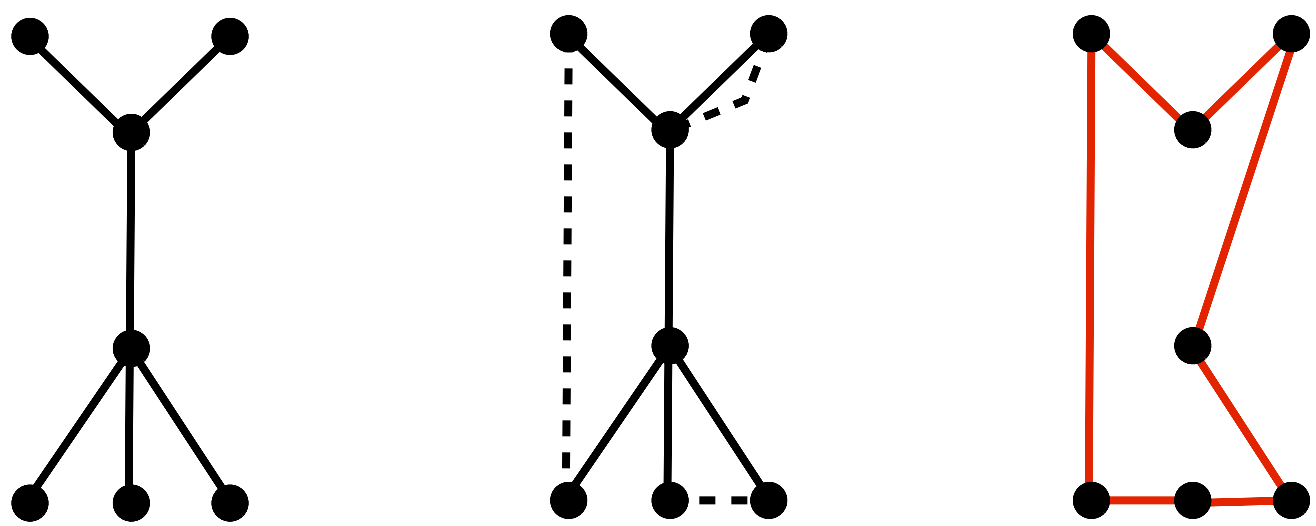
Christofides' Algorithm

Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



Christofides' Algorithm

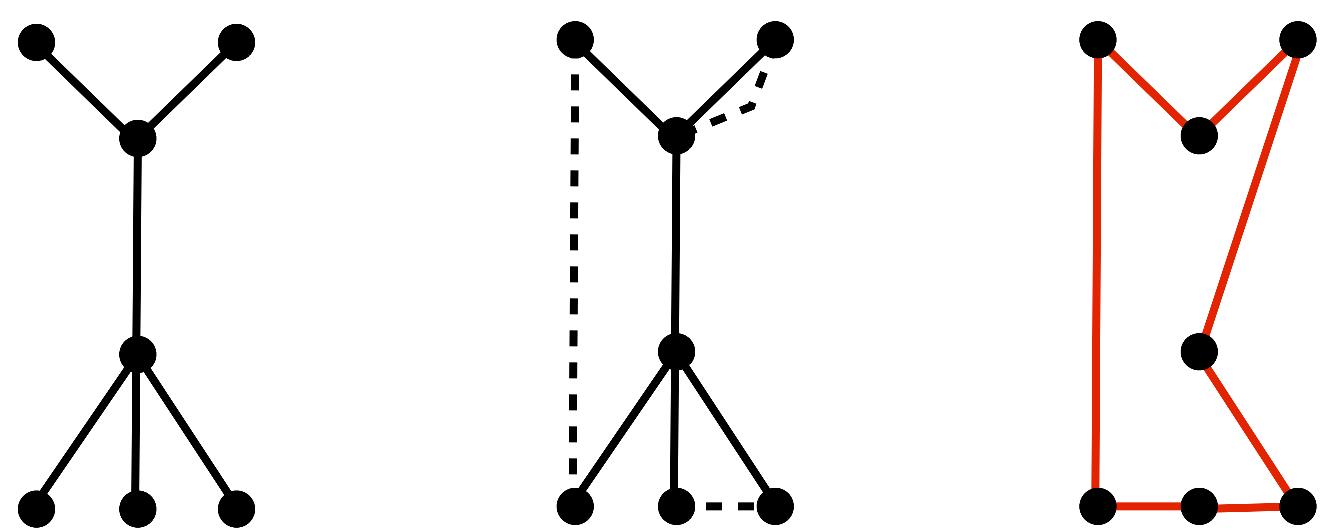
Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



$\leq$ OPT(c) +

Christofides' Algorithm

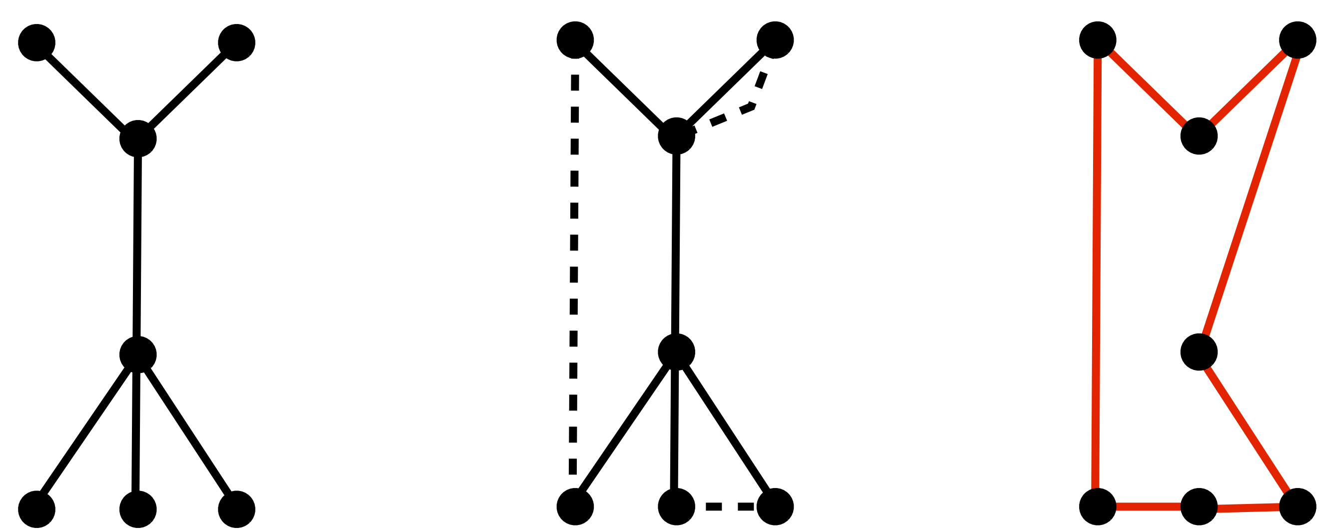
Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



$\leq$ OPT(c) + $\leq$ 1/2 OPT(c)

Christofides' Algorithm

Christofides (1976) shows how to compute a tour in polynomial time of cost 3/2 optimal: compute a min-cost spanning tree, compute a matching on the odd-degree vertices, then "shortcut" a traversal of the resulting Eulerian graph.



$\leq$ OPT(c)   + $\leq$ 1/2 OPT(c) $\leq$ 3/2 OPT(c)

An upper bound

- Wolsey (1980) and Shmoys and W (1990) show that OPT(c) can be replaced with SUBT(c), so that Christofides gives a tour of cost ≤ 3/2 SUBT(c).
- Therefore,

$$OPT(c) \leq \frac{3}{2} SUBT(c) \quad \Rightarrow \quad \gamma \leq \frac{OPT(c)}{SUBT(c)} \leq \frac{3}{2}$$

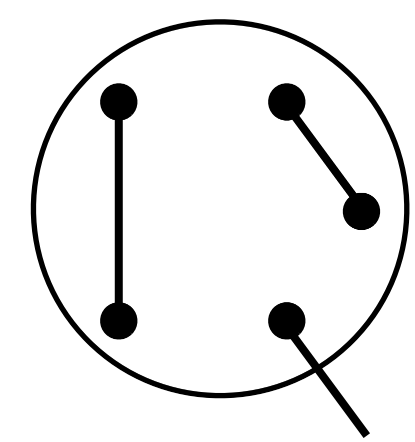
Perfect Matching Polytope

Edmonds (1965) shows that the min-cost perfect matching can be found as the solution to the linear program:

$$\text{Minimize} \sum_{e \in E} c(e) z(e)$$

$$\text{subject to} \sum_{e \in \delta(v)} z(e) = 1 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} z(e) \geq 1 \qquad \forall S \subset V, |S| \text{ odd}$$



Matchings and the Subtour LP

Then MATCH(c) $\leq$ 1/2 SUBT(c) since z = 1/2 x is feasible for the matching LP.

$$\text{Minimize} \sum_{e \in E} c(e)x(e)$$

subject to

$$\sum_{e \in \delta(v)} x(e) = 2 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} x(e) \geq 2 \qquad \forall S \subseteq V, |S| \geq 2$$

$$0 \leq x(e) \leq 1 \qquad \forall e \in E$$

$$\text{Minimize} \sum_{e \in E} c(e)z(e)$$

$$\text{subject to} \sum_{e \in \delta(v)} z(e) = 1 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} z(e) \geq 1 \qquad \forall S \subset V, |S| \text{ odd}$$

Shmoys and W (1990) also show that SUBT(c) is nonincreasing as vertices are removed so that matching on odd-degree vertices is at most 1/2 SUBT(c).

Recent results

- Some recent progress on graphic TSP (costs *c(i,j)* are the shortest *i-j* path distances in unweighted graph):
  - Boyd, Sitters, van der Ster, Stougie (2010): Gap is at most 4/3 if graph is cubic.
  - Oveis Gharan, Saberi, Singh (2010): Gap is at most 3/2 - ε for a constant ε > 0.
  - Mömke, Svensson (2011): Gap is at most 1.461.
  - Mömke, Svensson (2011): Gap is 4/3 if graph is subcubic (degree at most 3).
  - Mucha (2011): Gap is at most 13/9 ≈ 1.44.

Current state

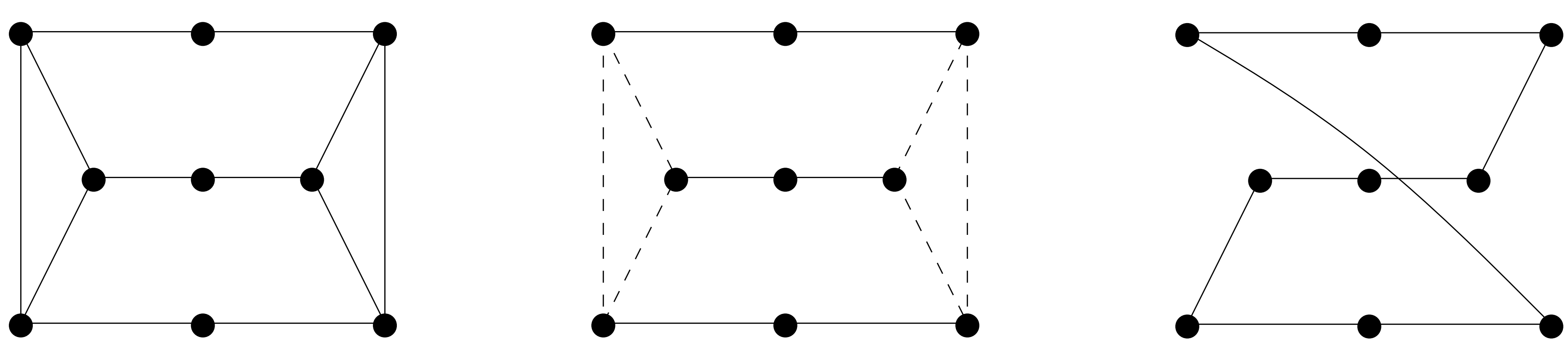
$$\frac{4}{3} \leq \gamma \leq \frac{3}{2}$$

- Conjecture (Goemans 1995, others): $\gamma = \dfrac{4}{3}$

More ignorance

Let γ_{12} be the integrality gap for costs $c(i,j) \in \{1,2\}$. Then all we know is

$$\frac{10}{9} \leq \gamma_{12} \leq \frac{3}{2}$$



cost 1 edges LP soln OPT

Still more ignorance

We don't even know the equivalent worst-case ratio between 2-matching costs 2M(c) and SUBT(c).

$$\mu \equiv \sup_n \mu(n) \text{ where } \mu(n) \equiv \sup_{c \in \mathcal{C}_n} \frac{2M(c)}{SUBT(c)}$$

Then all we know is that

$$\frac{10}{9} \le \mu \le \frac{4}{3} \text{ (Boyd, Carr 1999)}$$

Conjecture (Boyd, Carr 2011): $\mu = \frac{10}{9}$

Our contributions

- We can prove the Boyd-Carr conjecture.
- We can show $\gamma_{12} < 4/3$.

Outline

Outline

- $\mu \leq 4/3$ under a certain condition.

Outline

- $\mu \leq 4/3$ under a certain condition.
- $\mu \leq 10/9$ under the same condition.

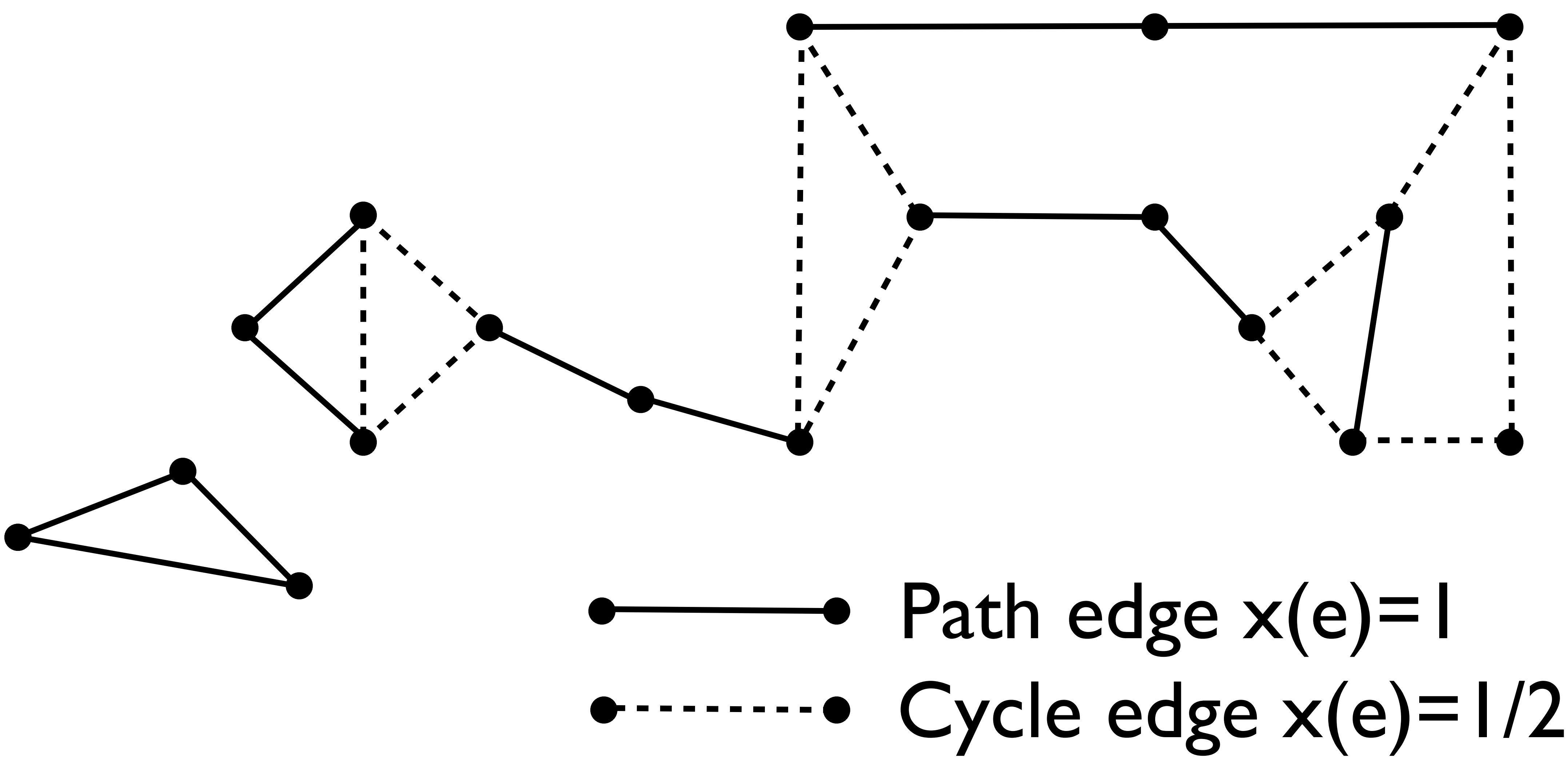
Outline

- $\mu \leq 4/3$ under a certain condition.
- $\mu \leq 10/9$ under the same condition.
- $\mu \leq 10/9$.

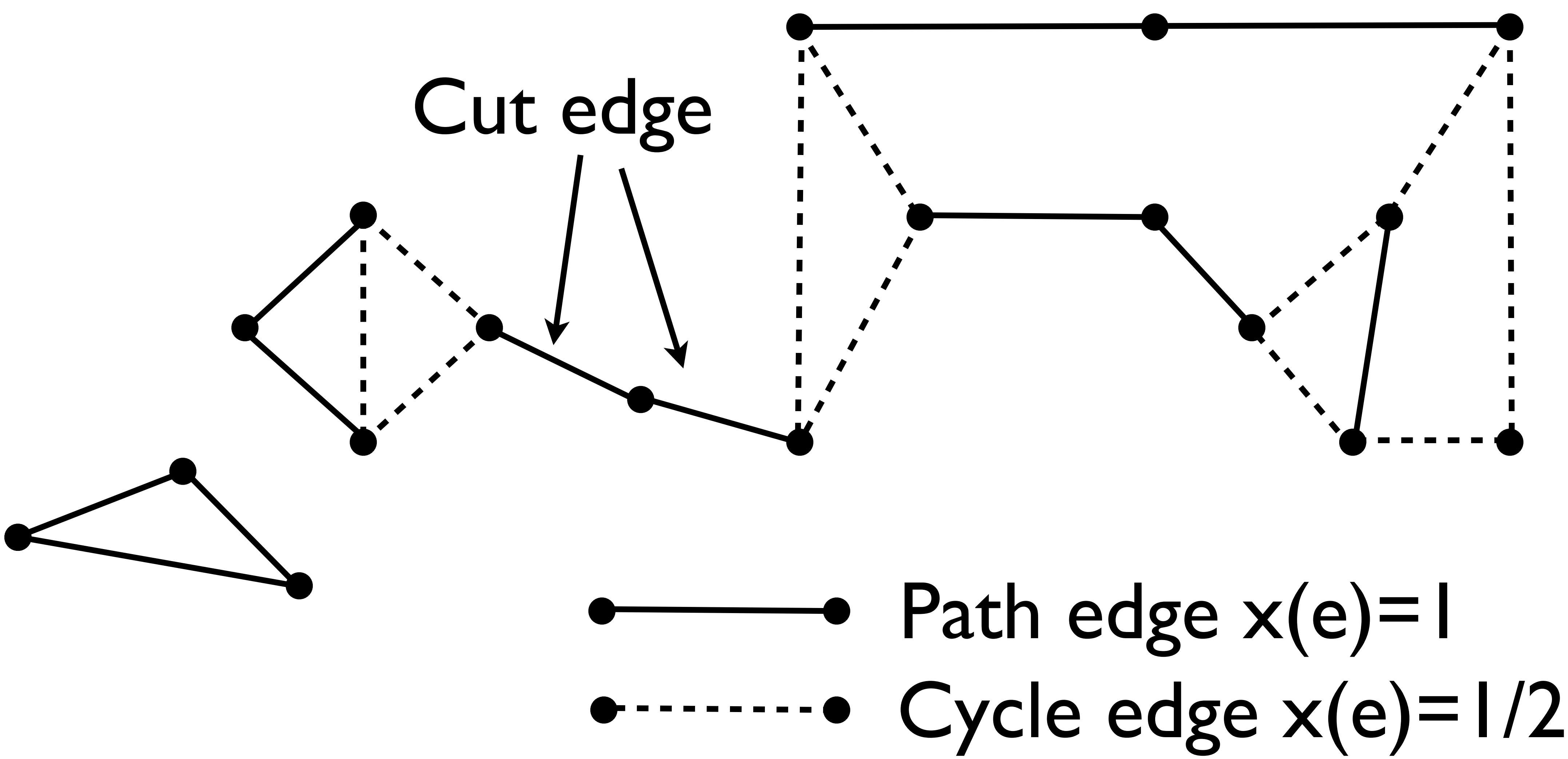
Outline

- $\mu \leq 4/3$ under a certain condition.
- $\mu \leq 10/9$ under the same condition.
- $\mu \leq 10/9$.
- Some conjectures.

Some terminology



Some terminology



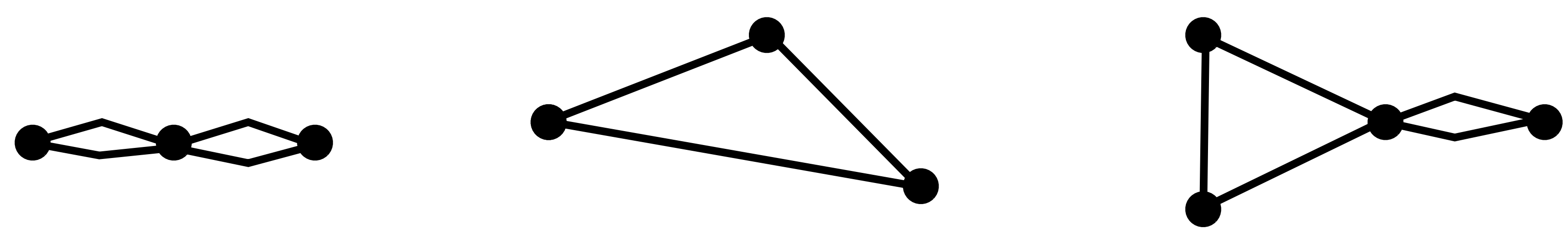
The strategy

The strategy

- Start with an optimal fractional 2-matching; this gives a lower bound on the Subtour LP.

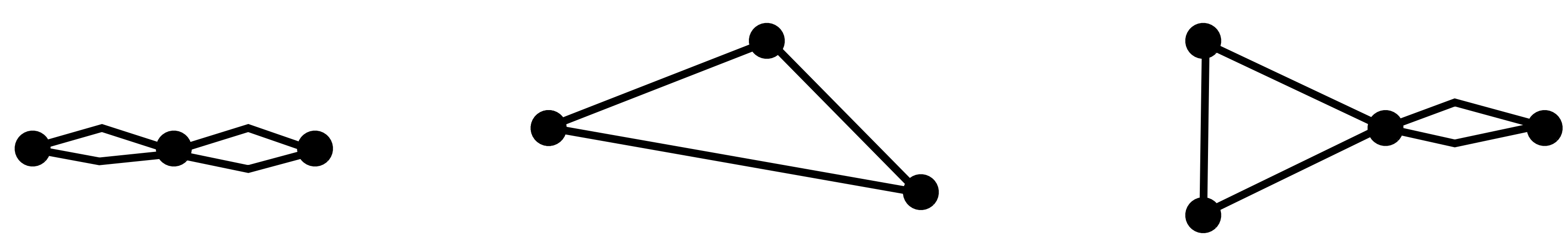
The strategy

- Start with an optimal fractional 2-matching; this gives a lower bound on the Subtour LP.
- Add a low-cost set of edges to create a *graphical* 2-matching: each vertex has degree 2 or 4; each component has size at least 3; each edge has 0, 1, or 2 copies.

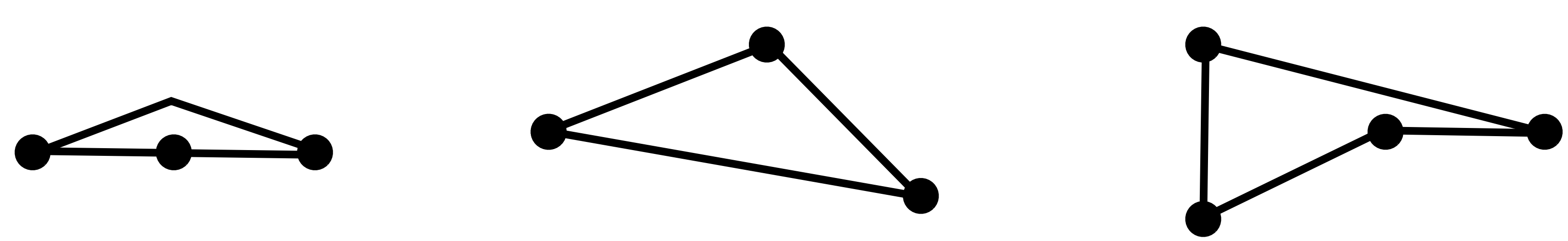


The strategy

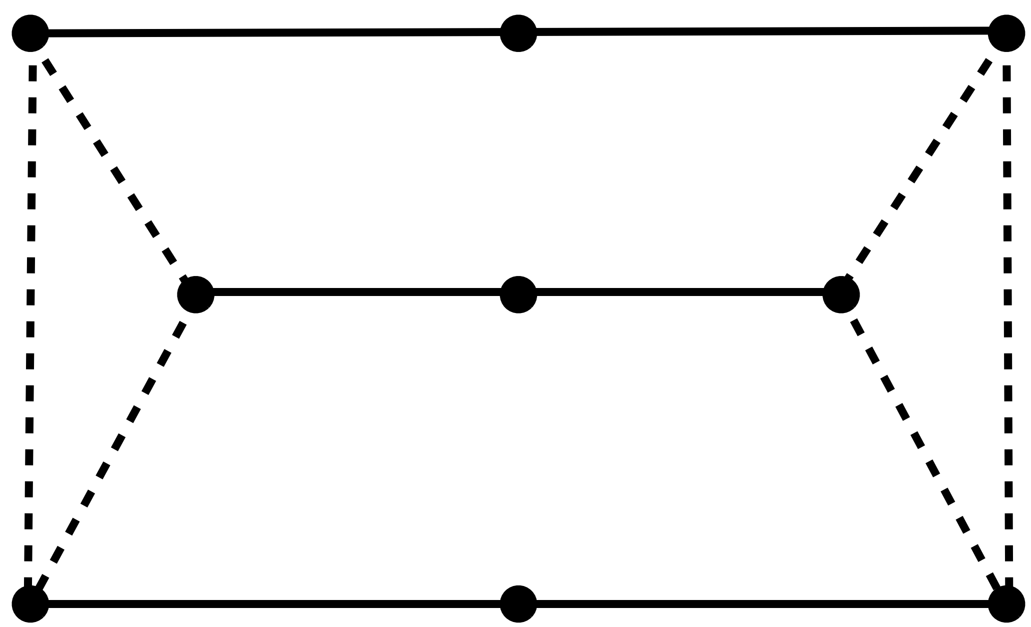
- Start with an optimal fractional 2-matching; this gives a lower bound on the Subtour LP.
- Add a low-cost set of edges to create a *graphical* 2-matching: each vertex has degree 2 or 4; each component has size at least 3; each edge has 0, 1, or 2 copies.



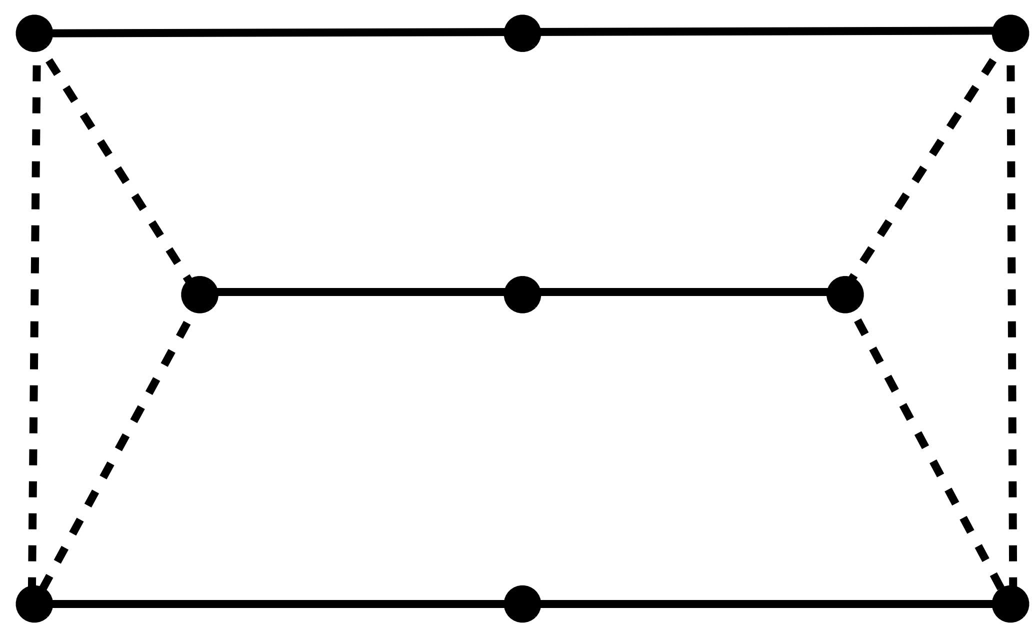
- "Shortcut" the graphical 2-matching to a 2-matching.



First consider fractional 2-matchings that have no cut edge, and show that we can get a graphical 2-matching with a 4/3 increase in cost.

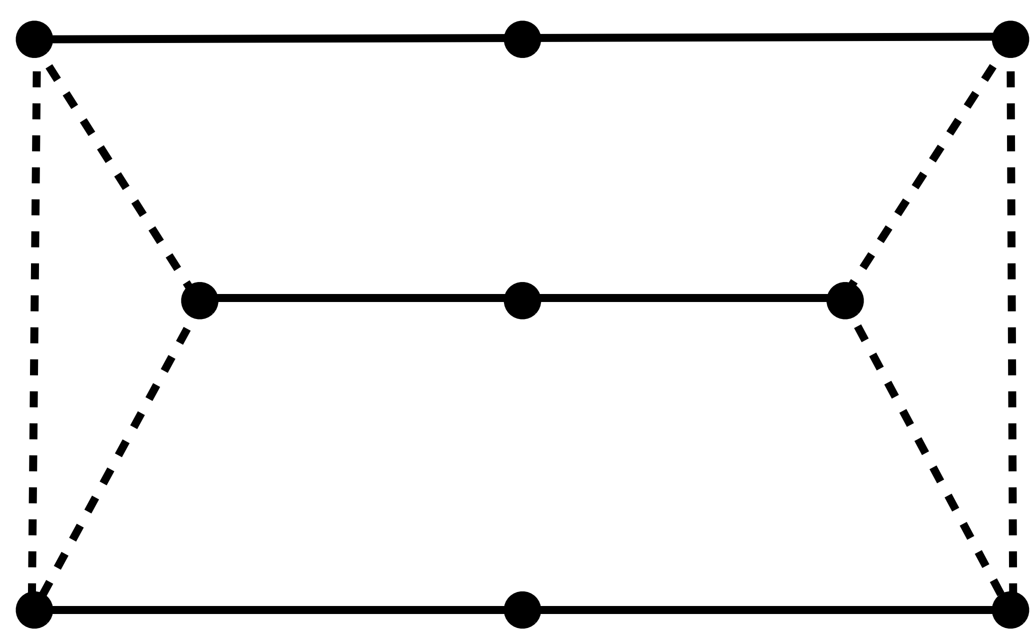


First consider fractional 2-matchings that have no cut edge, and show that we can get a graphical 2-matching with a 4/3 increase in cost.



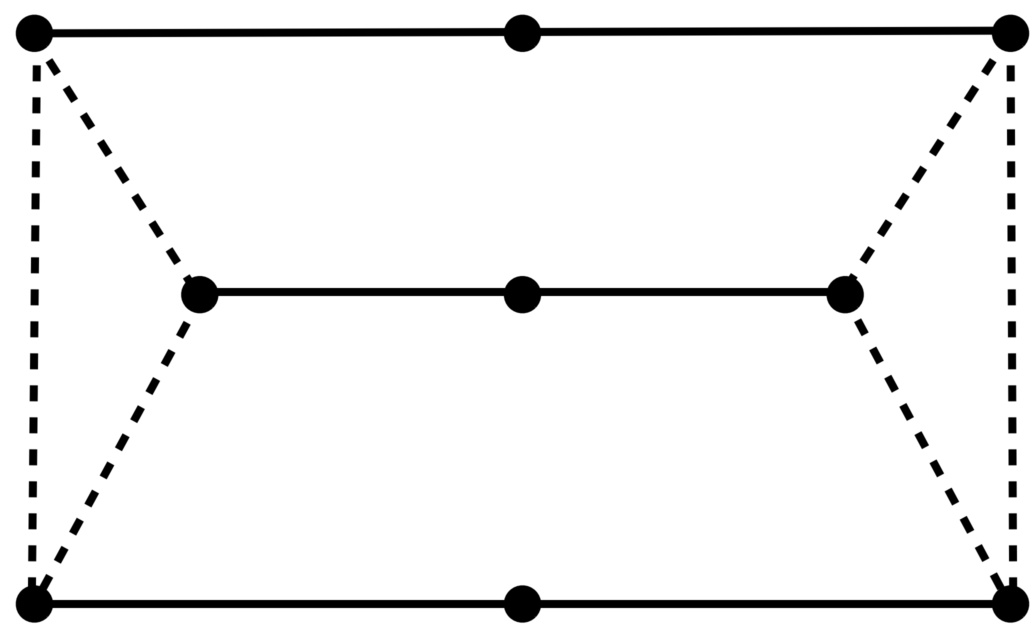
Graphical 2M $\leq$ 4/3 Fractional 2M

First consider fractional 2-matchings that have no cut edge, and show that we can get a graphical 2-matching with a 4/3 increase in cost.



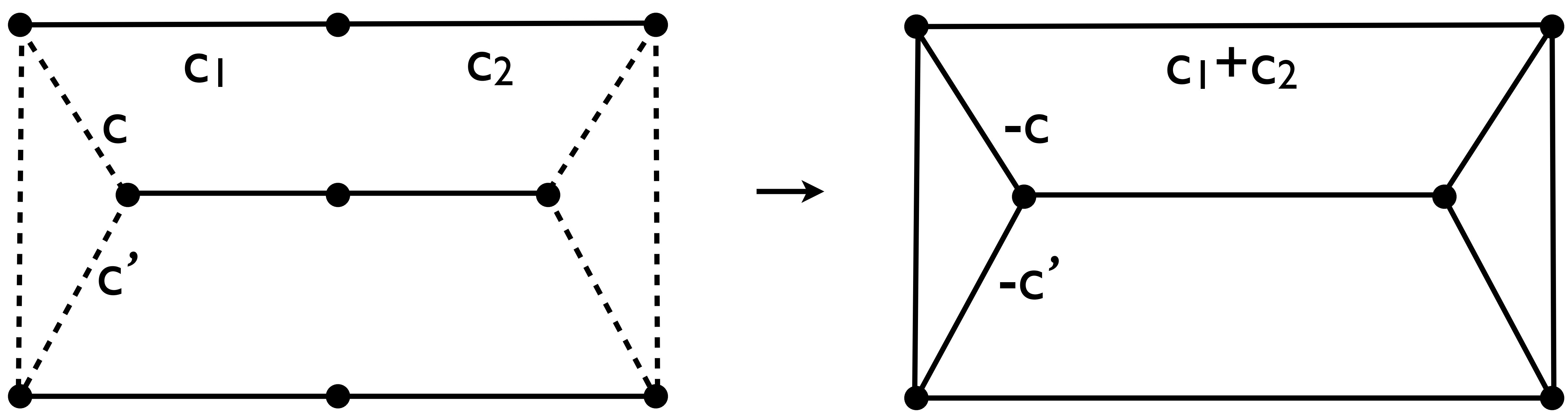
2M $\leq$ Graphical 2M $\leq$ 4/3 Fractional 2M

First consider fractional 2-matchings that have no cut edge, and show that we can get a graphical 2-matching with a 4/3 increase in cost.



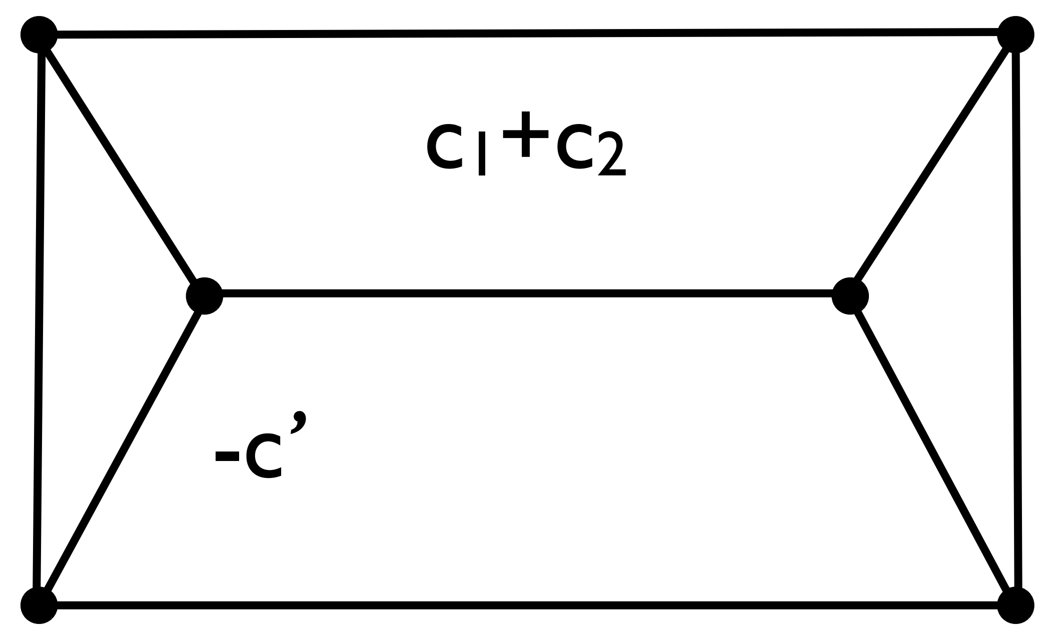
2M $\leq$ Graphical 2M $\leq$ 4/3 Fractional 2M $\leq$ 4/3 Subtour

Create new graph by replacing path edges with a single edge of cost equal to the path, cycle edges with negations of their cost.

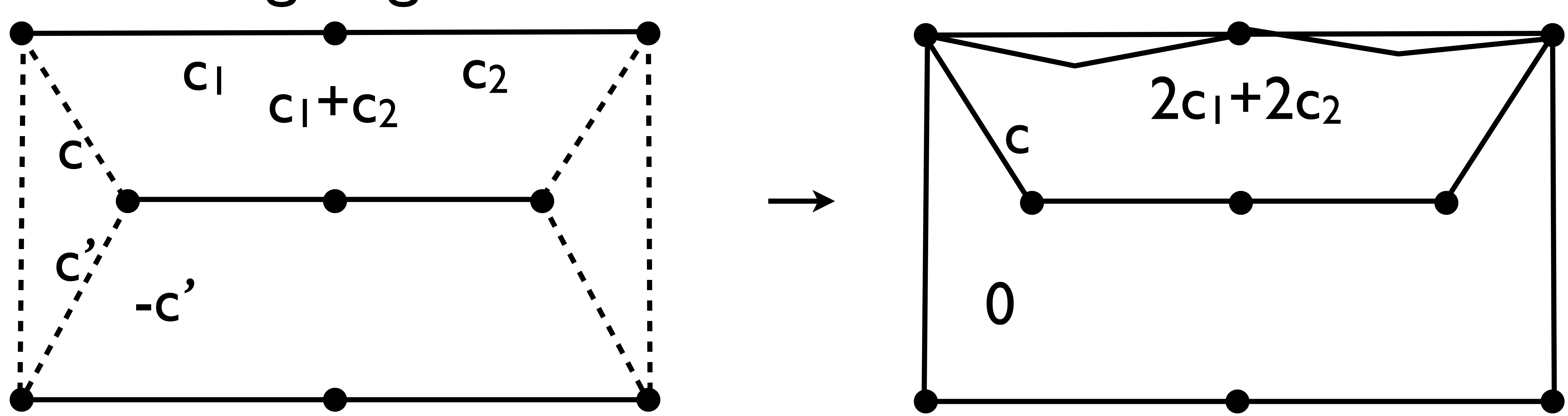


New graph is cubic and 2-edge connected.

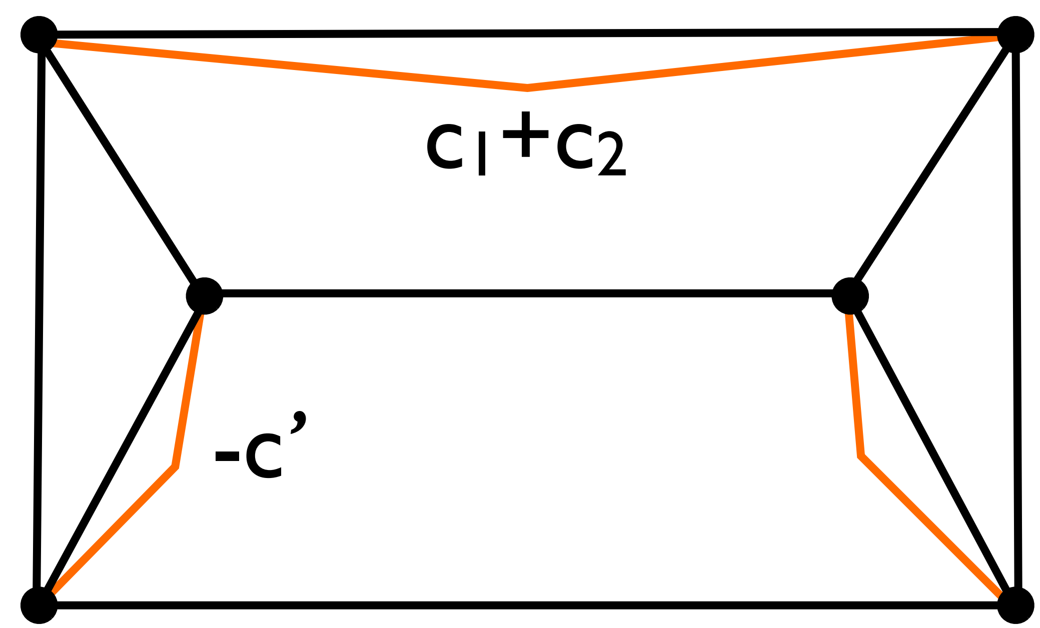
Compute a min-cost perfect matching in new graph.



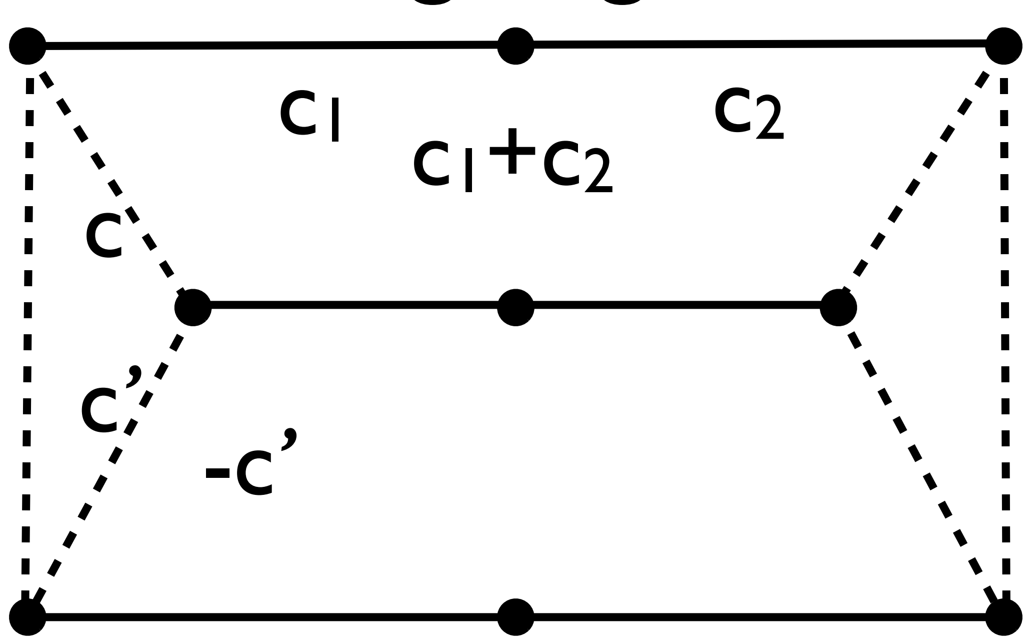
In the fractional 2-matching, double any path edge in matching, remove any cycle edge. Cost is paths + cycles + matching edges.



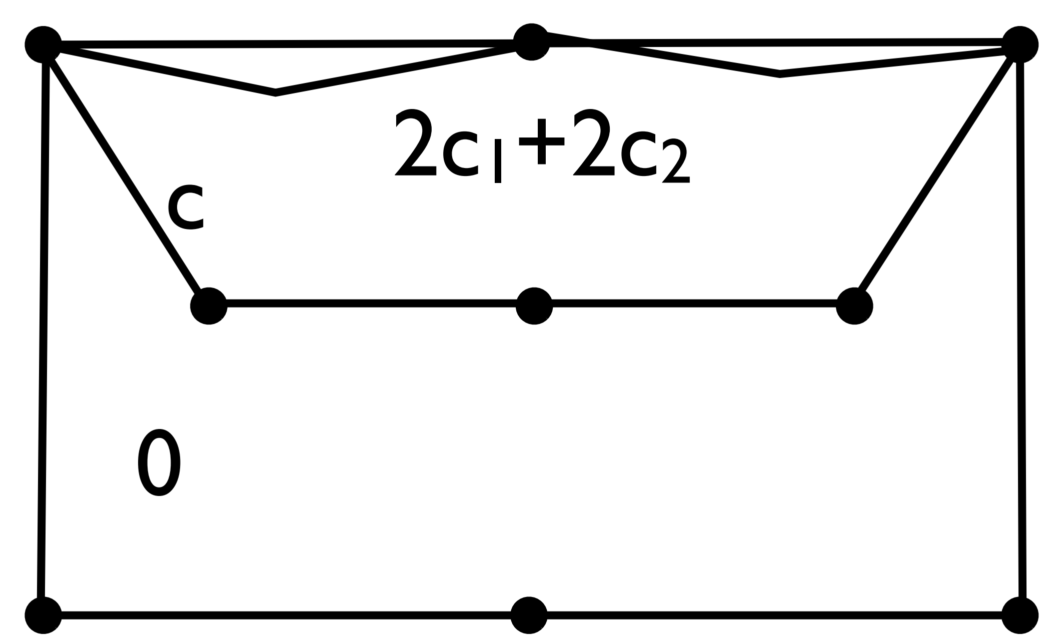
Compute a min-cost perfect matching in new graph.



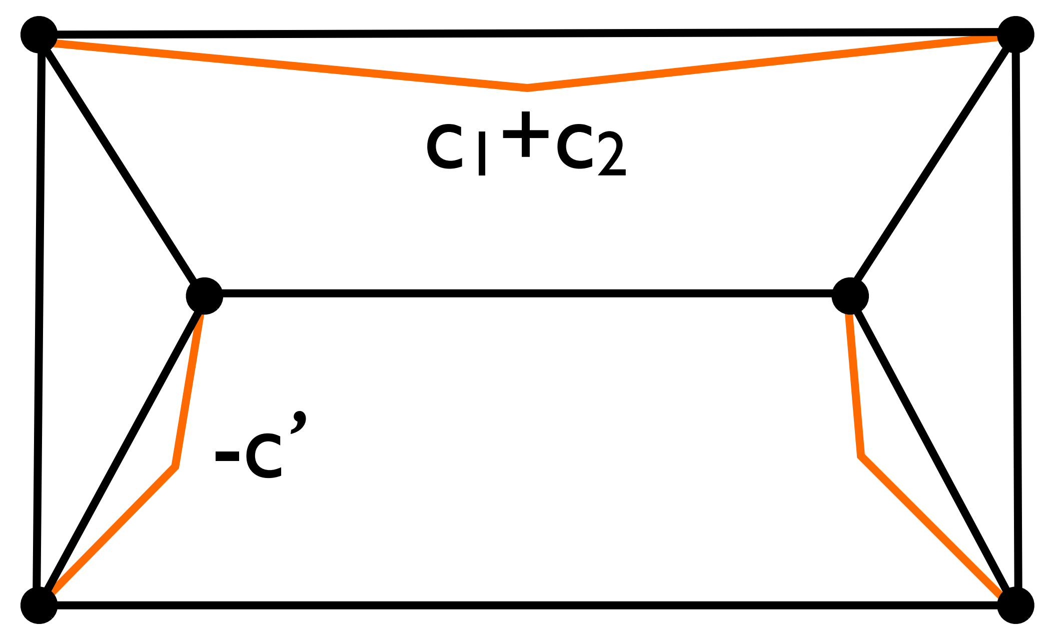
In the fractional 2-matching, double any path edge in matching, remove any cycle edge. Cost is paths + cycles + matching edges.



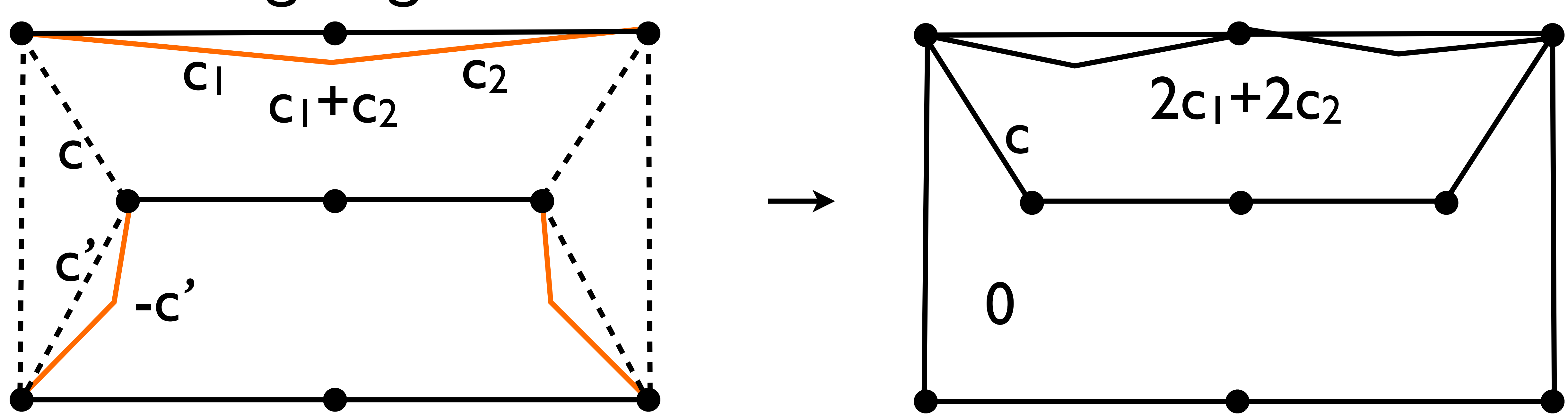
$\rightarrow$



Compute a min-cost perfect matching in new graph.

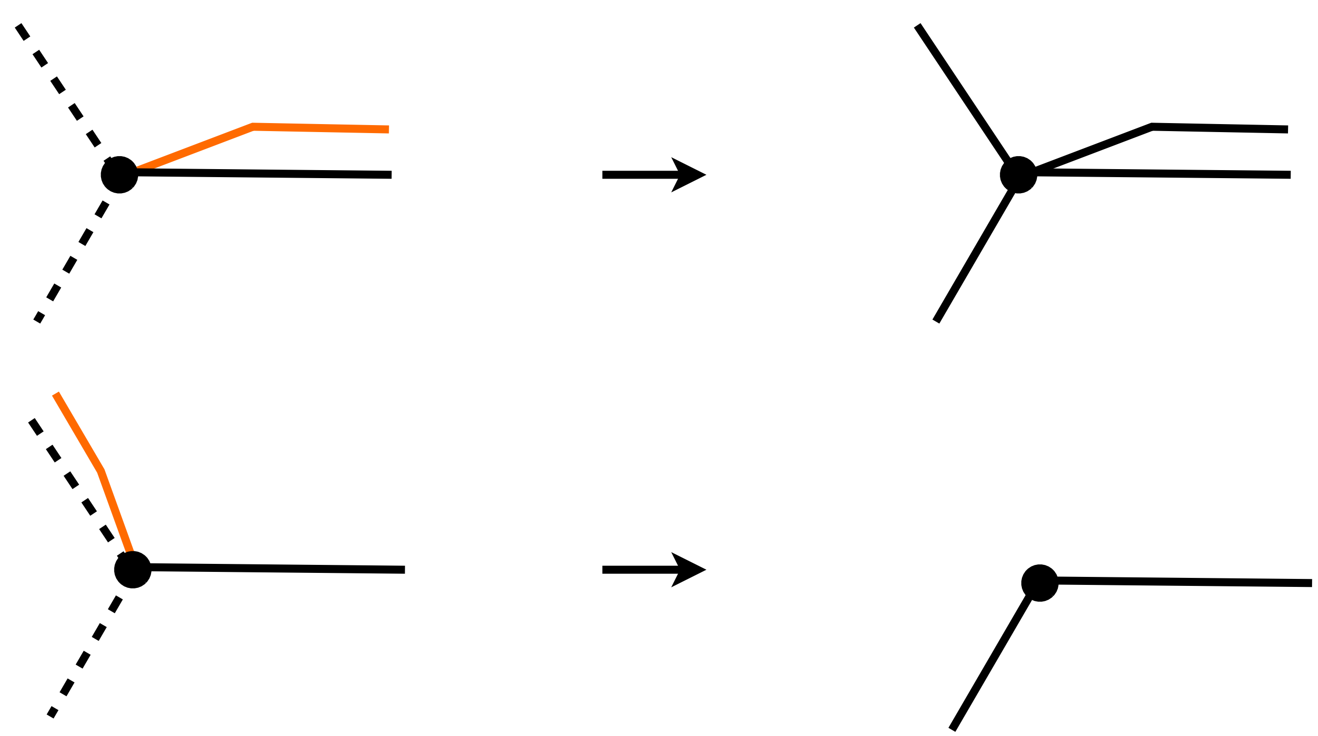


In the fractional 2-matching, double any path edge in matching, remove any cycle edge. Cost is paths + cycles + matching edges.



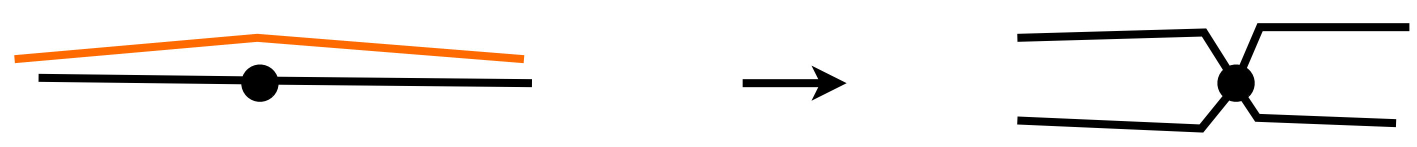
Why this works

For any given node on the cycle, either its associated path edge is in the matching or one of the two cycle edges.



Why this works

For any given node on the path, either its associated path edge is in the matching or not.



Bounding the cost

- P = total cost of all path edges
- C = total cost all cycle edges
- So fractional 2-matching costs P + C/2
- Claim: Perfect matching in the new graph costs at most 1/3 the cost of all its edges, so at most 1/3(P - C)

Bounding the cost

- Since the graphical 2-matching costs at most P + C + matching, it costs at most
$$P + C + \frac{1}{3}(P - C) = \frac{4}{3}P + \frac{2}{3}C = \frac{4}{3}\left(P + \frac{1}{2}C\right)$$

Bounding the cost

- Since the graphical 2-matching costs at most P + C + matching, it costs at most

$$P + C + \frac{1}{3}(P - C) = \frac{4}{3}P + \frac{2}{3}C = \frac{4}{3}\left(P + \frac{1}{2}C\right)$$

Graphical 2M ≤ 4/3 Fractional 2M

Bounding the cost

- Since the graphical 2-matching costs at most P + C + matching, it costs at most

$$P + C + \frac{1}{3}(P - C) = \frac{4}{3}P + \frac{2}{3}C = \frac{4}{3}\left(P + \frac{1}{2}C\right)$$

2M ≤ Graphical 2M ≤ 4/3 Fractional 2M

Bounding the cost

- Since the graphical 2-matching costs at most P + C + matching, it costs at most

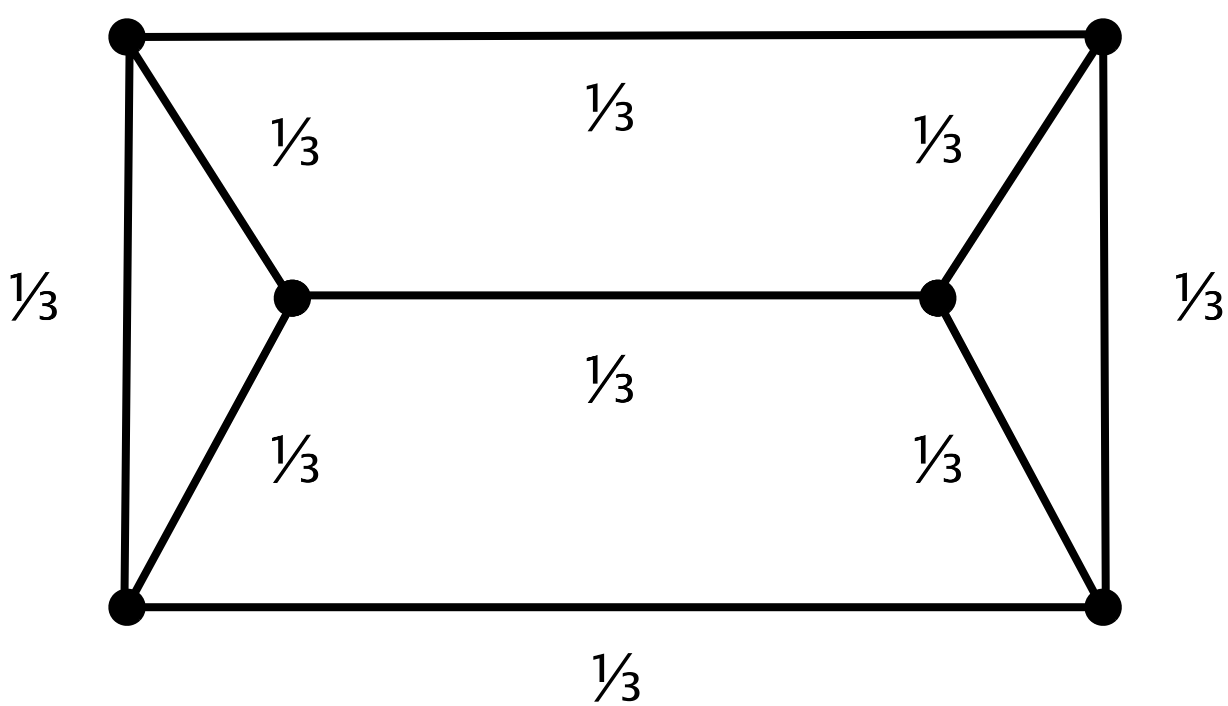
$$P + C + \frac{1}{3}(P - C) = \frac{4}{3}P + \frac{2}{3}C = \frac{4}{3}\left(P + \frac{1}{2}C\right)$$

2M ≤ Graphical 2M ≤ 4/3 Fractional 2M

≤ 4/3 Subtour

Matching cost

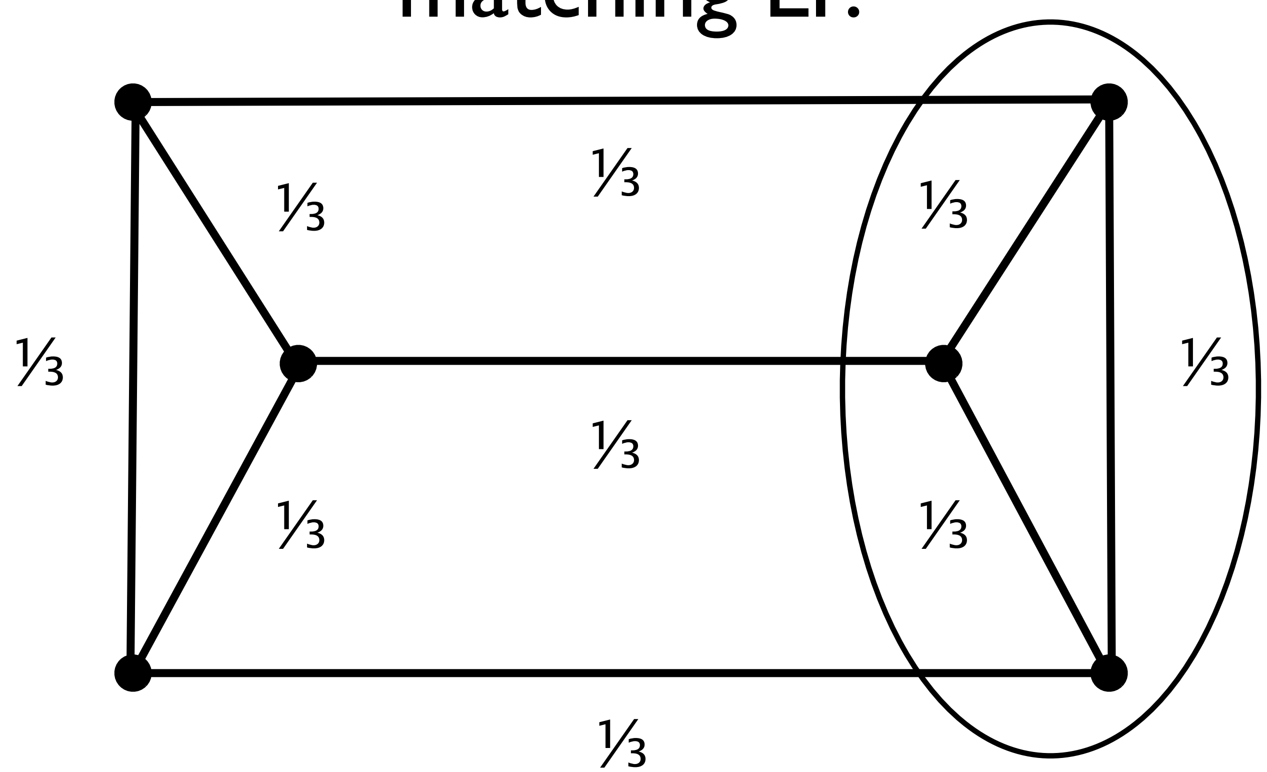
- Naddef and Pulleyblank (1981): Any cubic, 2-edge-connected, weighted graph has a perfect matching of cost at most a third of the sum of the edge weights.
- Proof: Set z(e)=1/3 for all e∈E, then feasible for matching LP.



$$
\begin{array}{llll}
\text{Minimize} & \displaystyle\sum_{e\in E} c(e)z(e) & & \\
\text{subject to} & \displaystyle\sum_{e\in\delta(v)} z(e) = 1 & & \forall v \in V \\
 & \displaystyle\sum_{e\in\delta(S)} z(e) \geq 1 & & \forall S \subset V, |S| \text{ odd}
\end{array}
$$

Matching cost

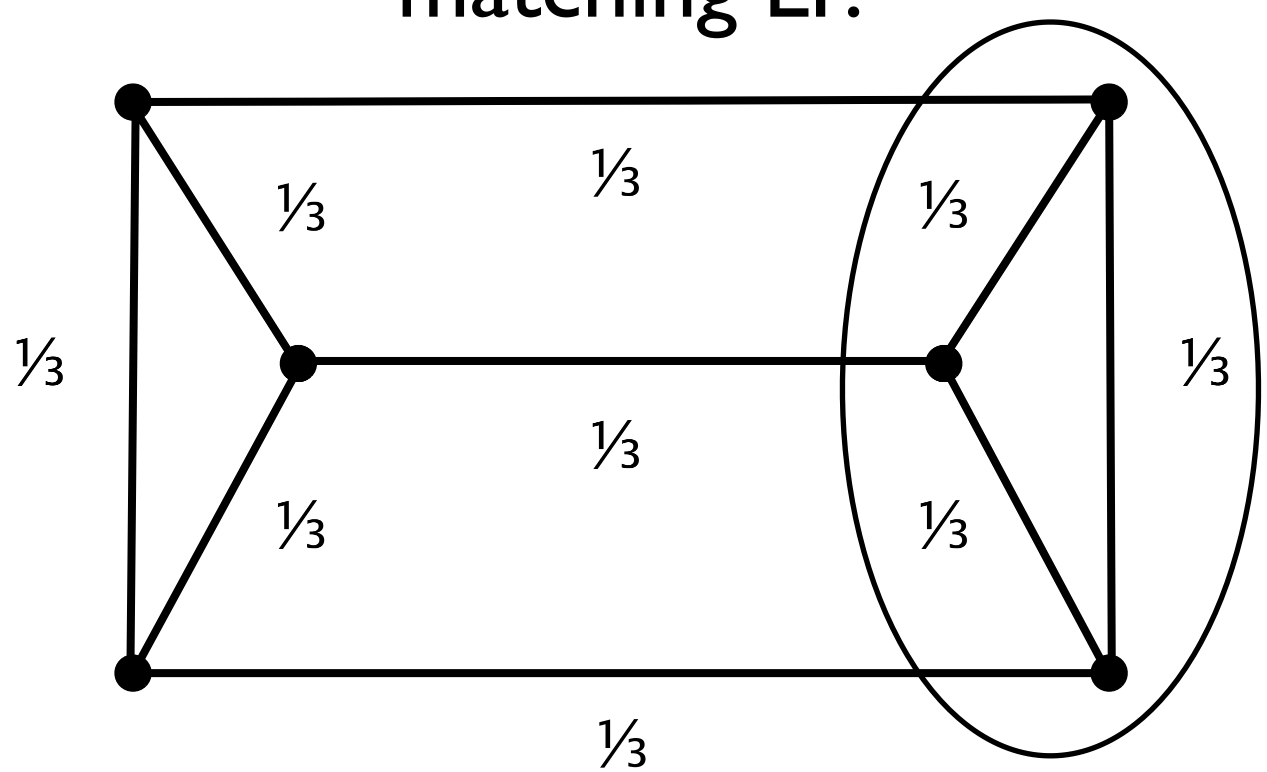
- Naddef and Pulleyblank (1981): Any cubic, 2-edge-connected, weighted graph has a perfect matching of cost at most a third of the sum of the edge weights.
- Proof: Set z(e)=1/3 for all e∈E, then feasible for matching LP.



$$
\begin{aligned}
\text{Minimize} \quad & \sum_{e \in E} c(e) z(e) \\
\text{subject to} \quad & \sum_{e \in \delta(v)} z(e) = 1 \qquad \forall v \in V \\
& \sum_{e \in \delta(S)} z(e) \geq 1 \qquad \forall S \subset V, |S| \text{ odd}
\end{aligned}
$$

Matching cost

- Naddef and Pulleyblank (1981): Any cubic, 2-edge-connected, weighted graph has a perfect matching of cost at most a third of the sum of the edge weights.
- Proof: Set z(e)=1/3 for all e∈E, then feasible for matching LP.

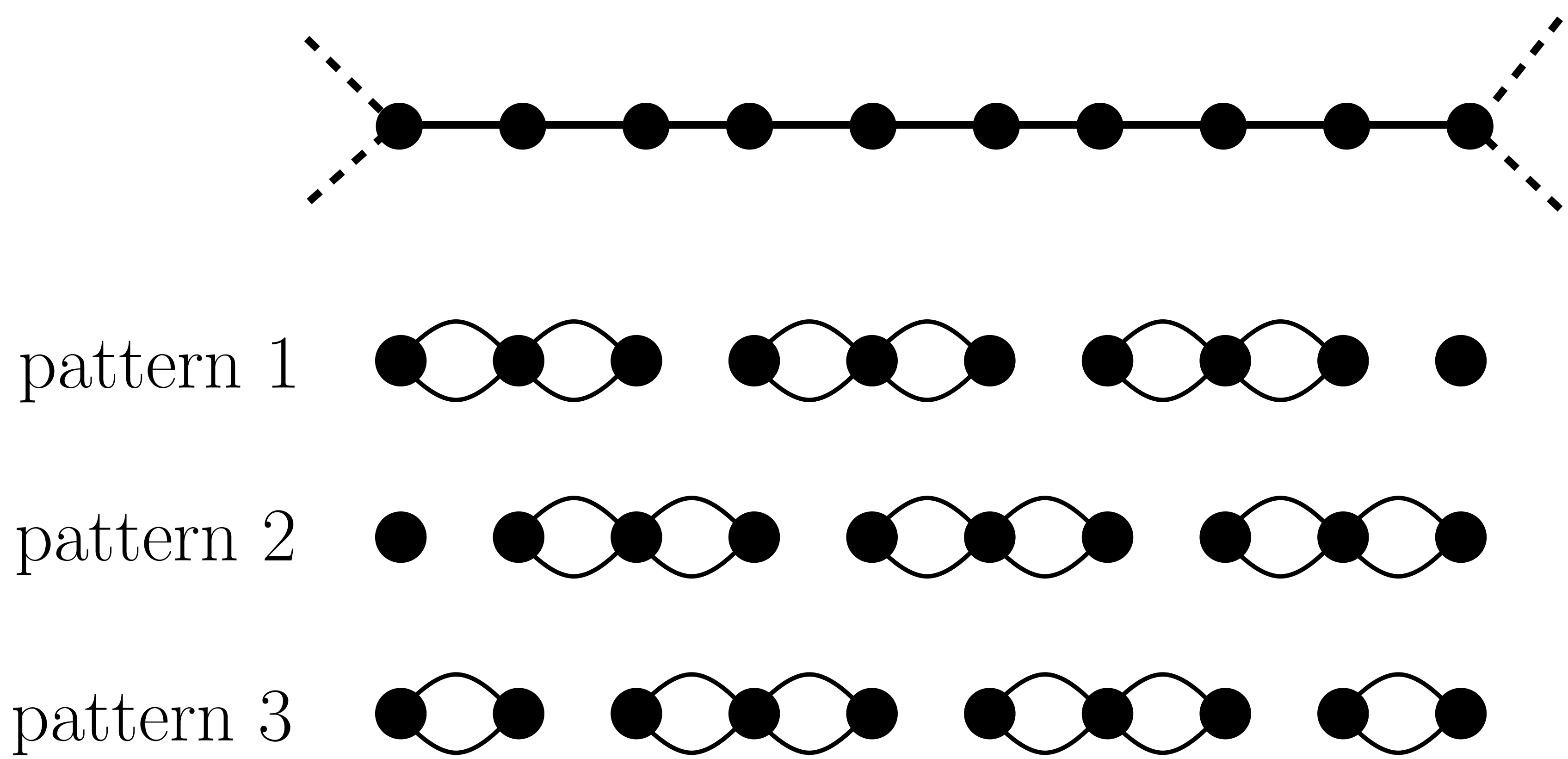


$$\begin{aligned} \text{Minimize} \quad & \sum_{e \in E} c(e) z(e) & \\ \text{subject to} \quad & \sum_{e \in \delta(v)} z(e) = 1 & \forall v \in V \\ & \sum_{e \in \delta(S)} z(e) \geq 1 & \forall S \subset V, |S| \text{ odd} \end{aligned}$$

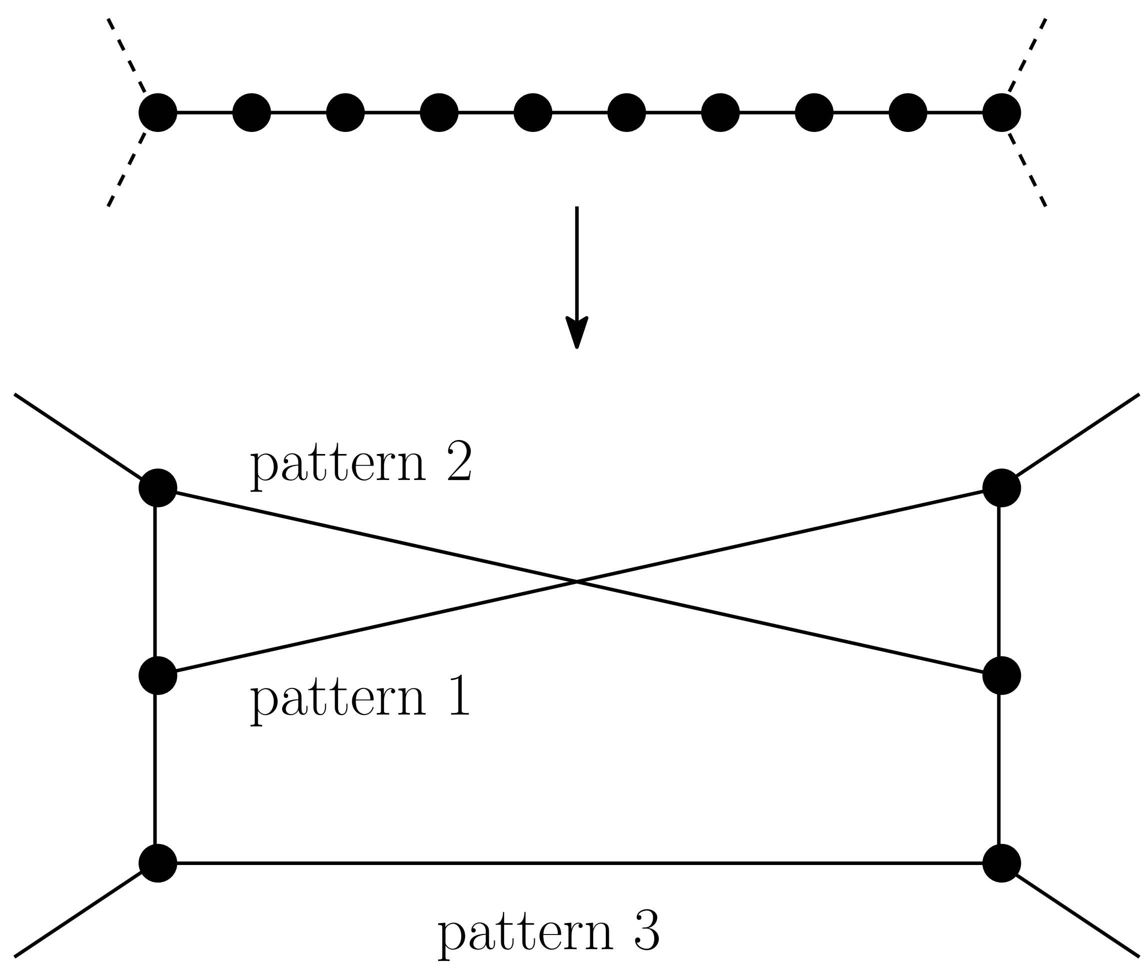
By parity argument any odd-sized set S must have odd |δ(S)|.

How to do better

Idea of Boyd and Carr (1999): Instead of duplicating an entire path, consider *patterns*.



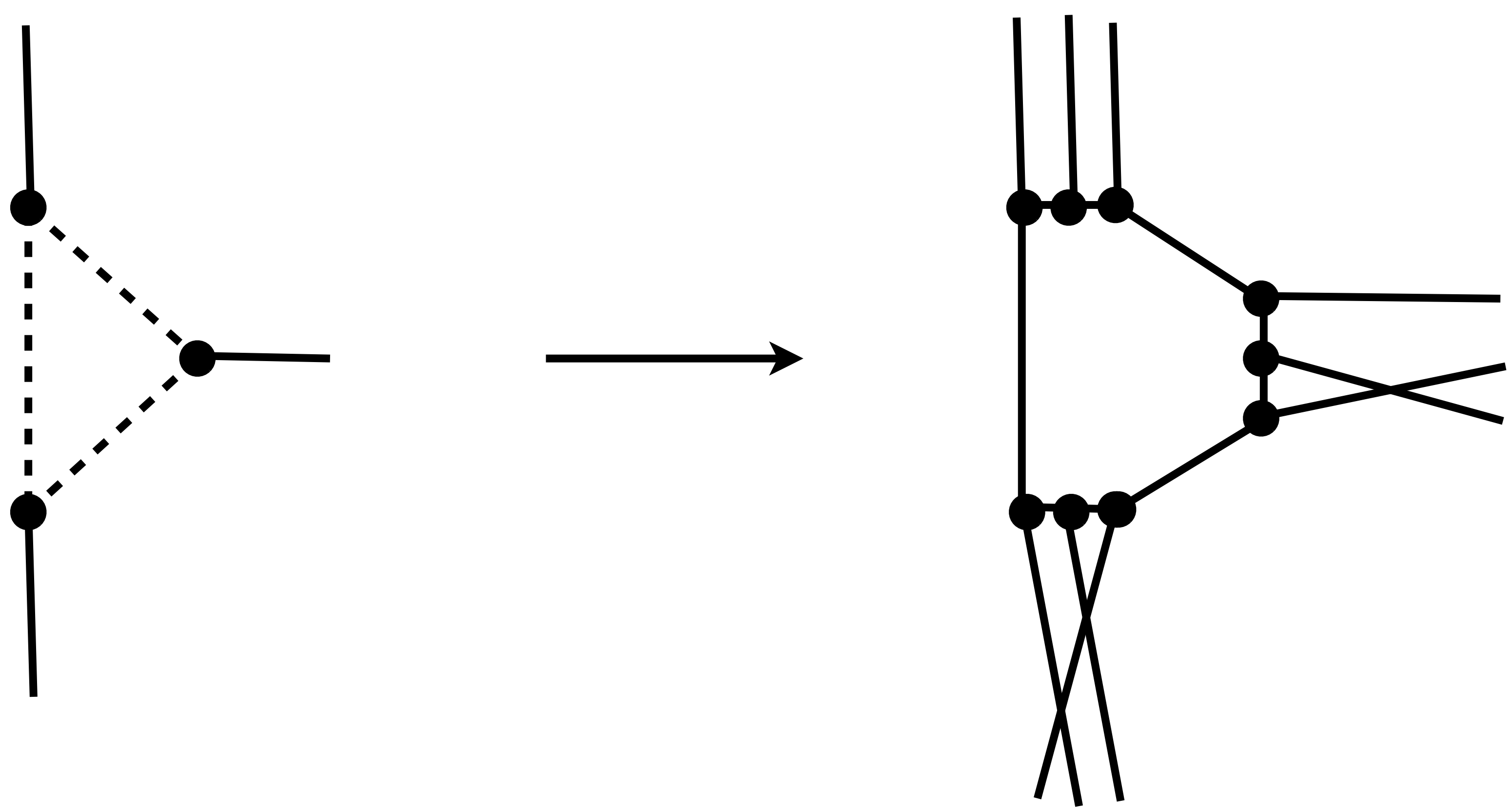
In new graph, replace every path with a *pattern gadget*; if the corresponding edge is in the matching, then we will use that pattern in the 2-matching.



Cost of pattern edge is difference in cost between pattern and path; other new edges have cost 0

Why does this help?

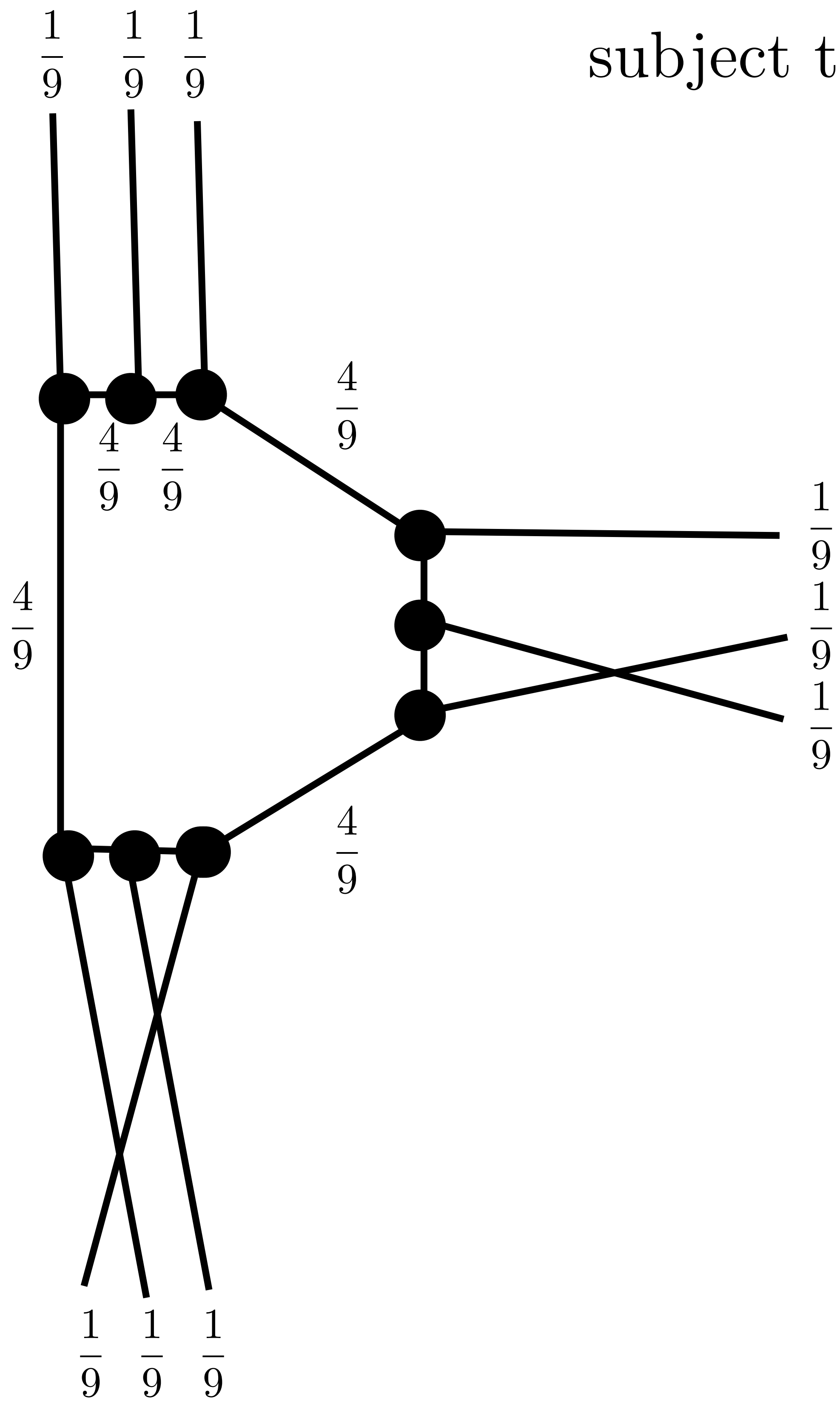
Intuition: Now we can get a cheaper matching.



$$\text{Minimize} \sum_{e \in E} c(e) z(e)$$

$$\text{subject to} \sum_{e \in \delta(v)} z(e) = 1 \qquad \forall v \in V$$

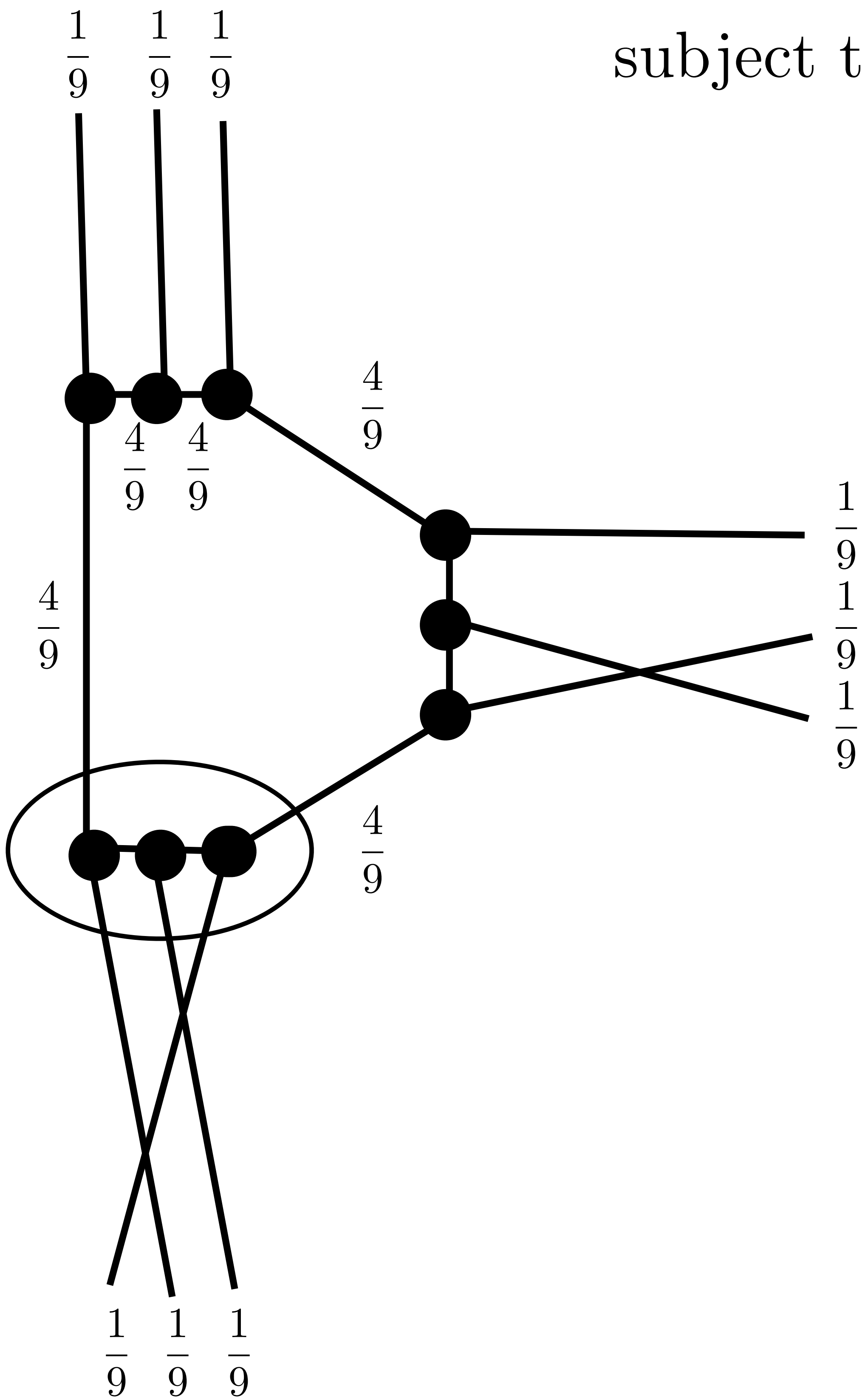
$$\sum_{e \in \delta(S)} z(e) \geq 1 \qquad \forall S \subset V, |S| \text{ odd}$$



- If any cycle edge in the cut, then at least two plus one more by parity: 4/9 + 4/9 + 1/9

- If no cycle edge in the cut, then at least 9 pattern edges.

- Can show matching has cost at most 1/9 P - 4/9 C



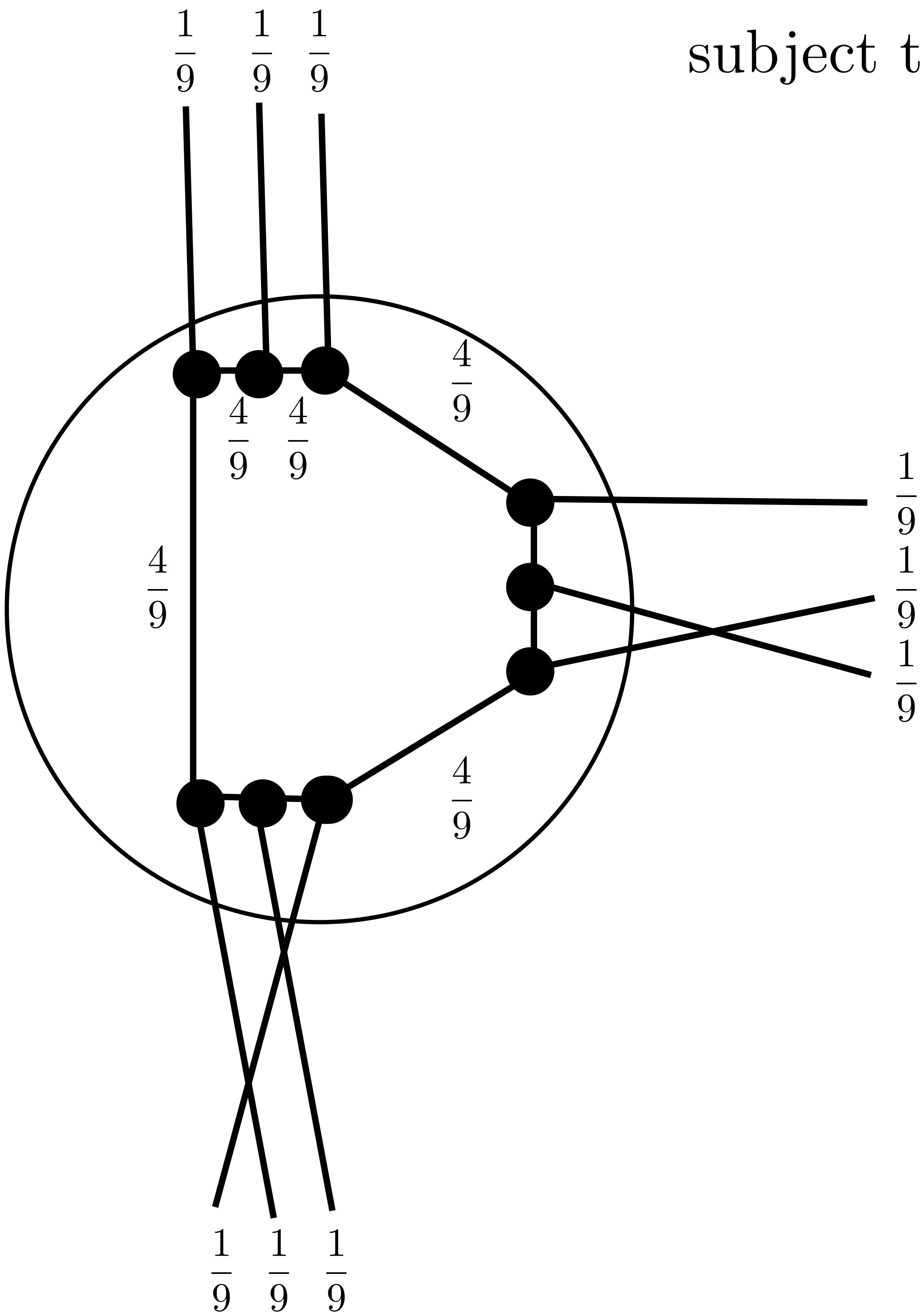
$$
\begin{aligned}
\text{Minimize} \quad & \sum_{e \in E} c(e) z(e) \\
\text{subject to} \quad & \sum_{e \in \delta(v)} z(e) = 1 \qquad \forall v \in V \\
& \sum_{e \in \delta(S)} z(e) \geq 1 \qquad \forall S \subset V, |S| \text{ odd}
\end{aligned}
$$

- If any cycle edge in the cut, then at least two plus one more by parity: 4/9 + 4/9 + 1/9
- If no cycle edge in the cut, then at least 9 pattern edges.
- Can show matching has cost at most 1/9 P - 4/9 C

$$\text{Minimize} \sum_{e \in E} c(e) z(e)$$

$$\text{subject to} \sum_{e \in \delta(v)} z(e) = 1 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} z(e) \geq 1 \qquad \forall S \subset V, |S| \text{ odd}$$



- If any cycle edge in the cut, then at least two plus one more by parity: 4/9 + 4/9 + 1/9
- If no cycle edge in the cut, then at least 9 pattern edges.
- Can show matching has cost at most 1/9 P - 4/9 C

Bounding the cost

- P = total cost of all path edges
- C = total cost all cycle edges
- So fractional 2-matching costs P + C/2
- Perfect matching in the new graph costs at most 1/9 P - 4/9 C

Bounding the cost

- Can show again that the graphical 2-matching costs at most P + C + matching, so it costs at most

$$P + C + \frac{1}{9}P - \frac{4}{9}C = \frac{10}{9}P + \frac{5}{9}C = \frac{10}{9}\left(P + \frac{1}{2}C\right)$$

Bounding the cost

- Can show again that the graphical 2-matching costs at most P + C + matching, so it costs at most

$$P + C + \frac{1}{9}P - \frac{4}{9}C = \frac{10}{9}P + \frac{5}{9}C = \frac{10}{9}\left(P + \frac{1}{2}C\right)$$

Graphical 2M ≤ 10/9 Fractional 2M

Bounding the cost

- Can show again that the graphical 2-matching costs at most P + C + matching, so it costs at most

$$P + C + \frac{1}{9}P - \frac{4}{9}C = \frac{10}{9}P + \frac{5}{9}C = \frac{10}{9}\left(P + \frac{1}{2}C\right)$$

2M $\leq$ Graphical 2M $\leq$ 10/9 Fractional 2M

Bounding the cost

- Can show again that the graphical 2-matching costs at most P + C + matching, so it costs at most

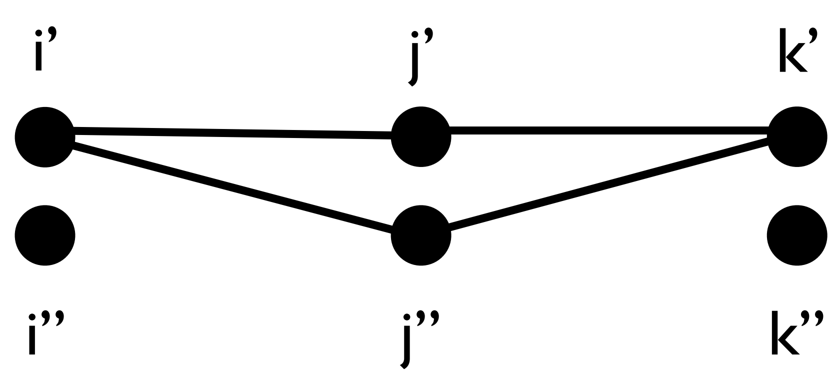
$$P + C + \frac{1}{9}P - \frac{4}{9}C = \frac{10}{9}P + \frac{5}{9}C = \frac{10}{9}\left(P + \frac{1}{2}C\right)$$

2M ≤ Graphical 2M ≤ 10/9 Fractional 2M

≤ 10/9 Subtour

Another route

- To prove stronger results, we give a polyhedral formulation for graphical 2-matchings.
- For all i∈V, create i' and i''
  - i' *required*: must have degree 2
  - i'' *optional*: may have degree 0 or 2
- For all (i,j)∈E, create edges (i',j'), (i',j''), (i'',j')



The formulation

$$\sum_{e \in \delta(i')} y(e) = 2 \qquad \forall i'$$

$$\sum_{e \in \delta(i'')} y(e) \leq 2 \qquad \forall i''$$

$$\sum_{e \in \delta(S) - F} y(e) + |F| - \sum_{e \in F} y(e) \geq 1 \qquad \forall S \subseteq V, F \subseteq \delta(S), F \text{ matching}, |F| \text{ odd}$$

$$0 \leq y(e) \leq 1 \qquad \forall e \in E$$

Showing that μ ≤10/9

Given Subtour LP soln x, set

$$y(i',j') = \frac{8}{9}x(i,j)$$
$$y(i'',j') = \frac{1}{9}x(i,j)$$
$$y(i',j'') = \frac{1}{9}x(i,j)$$

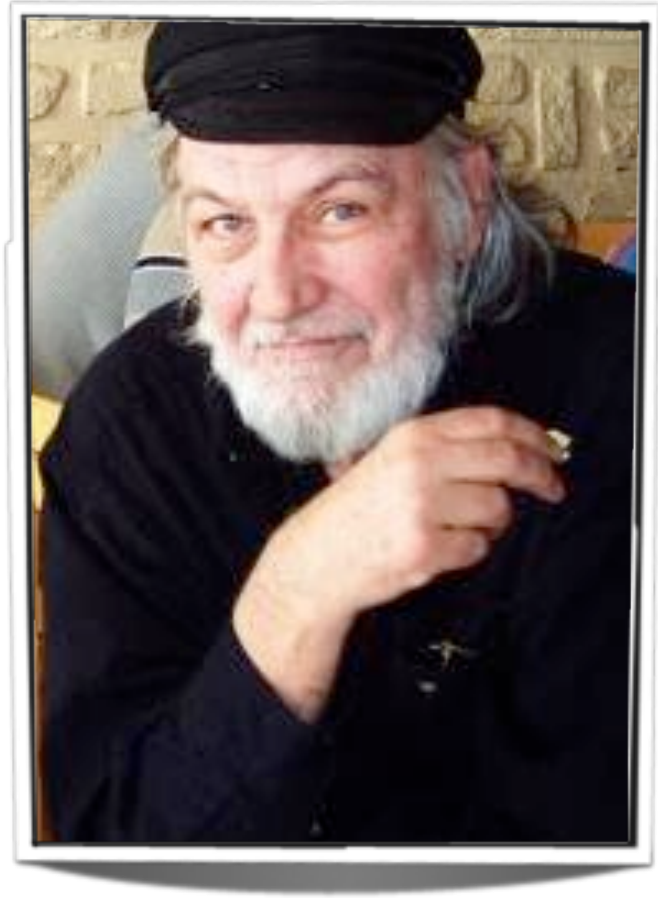
$$\sum_{e\in\delta(i')} y(e) = 2 \qquad \forall i'$$
$$\sum_{e\in\delta(i'')} y(e) \leq 2 \qquad \forall i''$$
$$\sum_{e\in\delta(S)-F} y(e) + |F| - \sum_{e\in F} y(e) \geq 1$$
$$\forall S \subseteq V, F \subseteq \delta(S), F \text{ matching}, |F| \text{ odd}$$
$$0 \leq y(e) \leq 1 \qquad \forall e \in E$$

Minimize $\sum_{e\in E} c(e)x(e)$

subject to

$$\sum_{e\in\delta(v)} x(e) = 2 \qquad \forall v \in V$$
$$\sum_{e\in\delta(S)} x(e) \geq 2 \qquad \forall S \subseteq V, |S| \geq 2$$
$$0 \leq x(e) \leq 1 \qquad \forall e \in E$$

Edmonds (1967)



Edmonds (1967)

traveling saleman problem [cf. 4]. I conjecture that there is no good algorithm for the traveling saleman problem. My reasons are the same as for any mathematical conjecture: (1) It is a legitimate mathematical possibility, and (2) I do not know.

A good algorithm is known for finding, in any graph

Some conjectures

Some conjectures

- For the 1,2-TSP I conjecture that $\gamma_{12} = 10/9$. We show $\gamma_{12} \leq 106/81 \approx 1.31$.

Some conjectures

- For the 1,2-TSP I conjecture that $\gamma_{12} = 10/9$. We show $\gamma_{12} \leq 106/81 \approx 1.31$.
- Computation shows the conjecture is true for $n \leq 12$.

An observation

- We know
  - $\dfrac{2M(c)}{F2M(c)} \le \dfrac{4}{3}$ (Boyd, Carr 1999)
  - $\dfrac{2M(c)}{F2M(c)} \le \dfrac{10}{9} \quad \forall c \in \{1,2\}^n$ (this work)
- We conjecture $\gamma \le 4/3$, $\gamma_{12} \le 10/9$.
- Coincidence?

Final conjecture

Final conjecture

- Conjecture: The worst case for the Subtour LP integrality gap (both γ and γ_{12}) occurs for solutions that are fractional 2-matching.

Final conjecture

- Conjecture: The worst case for the Subtour LP integrality gap (both γ and γ_{12}) occurs for solutions that are fractional 2-matching.
- Note: we don't even know tight bounds on γ and γ_{12} in this case, though we can show $\gamma_{12} \leq 7/6$ in this case.


INI OPS stasjon
JAN MAYEN
TEORI ER NÅR MAN FORSTÅR ALT
MEN INGEN TING VIRKER
PRAKSIS ER NÅR ALT VIRKER
MEN INGEN FORSTÅR HVORFOR
PÅ DENNE STASJONEN FORENER VI TEORI OG PRAKSIS

“Theory is when we understand everything, but nothing works.

"Theory is when we understand everything, but nothing works.

Practice is when everything works, but we don't understand why.

"Theory is when we understand everything, but nothing works.

Practice is when everything works, but we don't understand why.

At this station, theory and practice are united, so that nothing works and no one understands why."

Thank you for your attention.