Baby steps towards TSP inapproximability

Michael Lampis
KTH Royal Institute of Technology

May 3, 2013

Acknowledgements

The material in this talk is based on the following papers:

- “Improved Inapproximability for TSP”, APPROX’12
- “New Inapproximability Bounds for TSP”, arxiv’13 (joint work with Marek Karpinski and Richard Schmied)

The Story

- The Traveling Salesman problem is famous and important. Unfortunately, it's NP-hard.
  - How well can we approximate it?
  - Big breakthroughs in algorithms recently. We set out to improve on inapproximability results.

The Story

- The Traveling Salesman problem is famous and important. Unfortunately, it's NP-hard.
  - How well can we approximate it?
  - Big breakthroughs in algorithms recently. We set out to improve on inapproximability results.

Main idea

- Hardness obtained through a reduction from a Constraint Satisfaction Problem (CSP)

The Story

- The Traveling Salesman problem is famous and important. Unfortunately, it's NP-hard.
  - How well can we approximate it?
  - Big breakthroughs in algorithms recently. We set out to improve on inapproximability results.

Main idea

- Reduction is easier if CSP has bounded # of occurrences

The Story

- The Traveling Salesman problem is famous and important. Unfortunately, it's NP-hard.
  - How well can we approximate it?
  - Big breakthroughs in algorithms recently. We set out to improve on inapproximability results.

Main idea

- We need inapproximability results for CSPs with bounded # of occurrences

The Story

- The Traveling Salesman problem is famous and important. Unfortunately, it's NP-hard.
  - How well can we approximate it?
  - Big breakthroughs in algorithms recently. We set out to improve on inapproximability results.

Main idea

- Such results use expander graphs

The Story

- The Traveling Salesman problem is famous and important. Unfortunately, it's NP-hard.
  - How well can we approximate it?
  - Big breakthroughs in algorithms recently. We set out to improve on inapproximability results.

Main idea

- Expander graphs $\rightarrow$
  $\rightarrow$Hardness for bounded occurrence CSPs $\rightarrow$
  $\rightarrow$Hardness for TSP

The Actual Story

Better Expanders

The Actual Story

Better Expanders

- A local improvement argument gives (slightly) better expander graphs than those already in the literature.

The Actual Story

Better Expanders

- A local improvement argument gives (slightly) better expander graphs than those already in the literature. ✓

See “Local Improvement Gives Better Expanders”, arxiv’12

The Actual Story

Better Expanders

- A local improvement argument gives (slightly) better expander graphs than those already in the literature. ✓

See “Local Improvement Gives Better Expanders”, arxiv’12

- But improvement is too small to matter!

The Actual Story

Second attempt:

- We will rely on amplifier graph constructions due to Berman and Karpinski.

The Actual Story

Second attempt:

- We will rely on amplifier graph constructions due to Berman and Karpinski.
- These will help us construct inapproximable CSPs with 3 or 5 occurrences for each variable.

The Actual Story

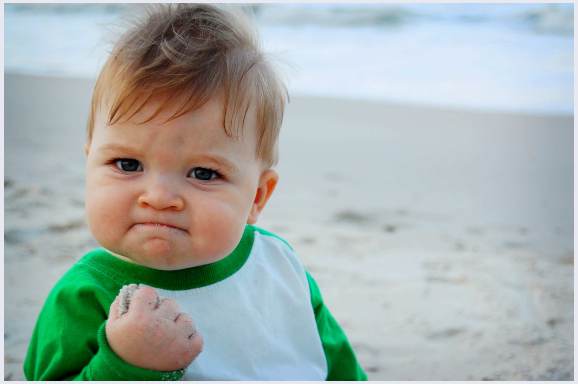
Second attempt:

- We will rely on amplifier graph constructions due to Berman and Karpinski.
- These will help us construct inapproximable CSPs with 3 or 5 occurrences for each variable.
- We will reduce these to TSP (and ATSP).

The Actual Story

Second attempt:

- We will rely on amplifier graph constructions due to Berman and Karpinski.
- These will help us construct inapproximable CSPs with 3 or 5 occurrences for each variable.
- We will reduce these to TSP (and ATSP).
- End result: simpler construction and better inapproximability constants!



The Actual Story

Second attempt:

- We will rely on amplifier graph constructions due to Berman and Karpinski.
- These will help us construct inapproximable CSPs with 3 or 5 occurrences for each variable.
- We will reduce these to TSP (and ATSP).
- End result: simpler construction and better inapproximability constants!

- Warning: don't expect a big improvement.

The Traveling Salesman Problem

The Traveling Salesman Problem

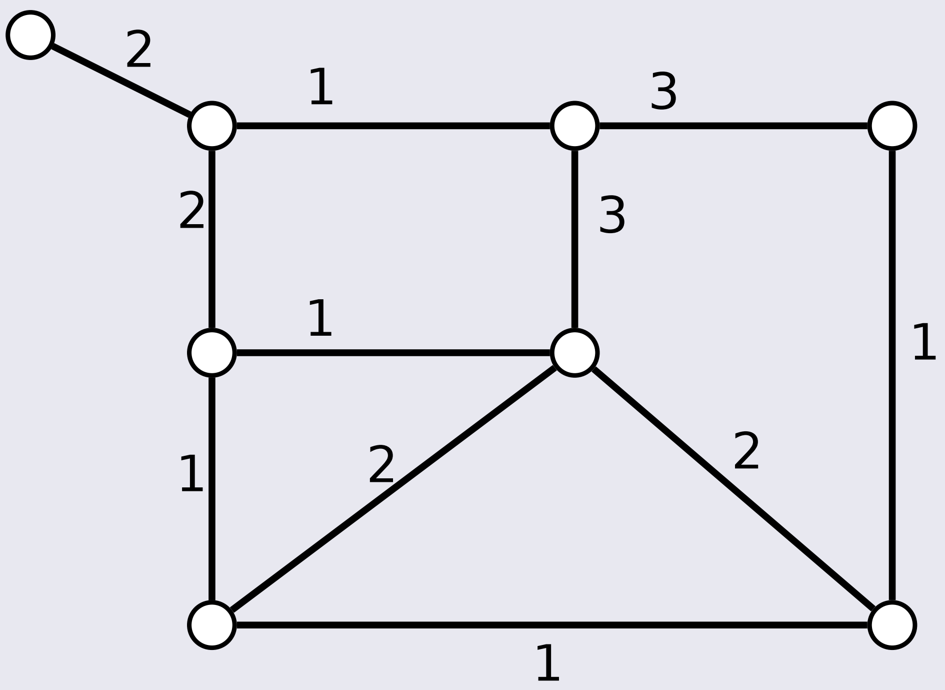
Input:

- An edge-weighted graph $G(V, E)$

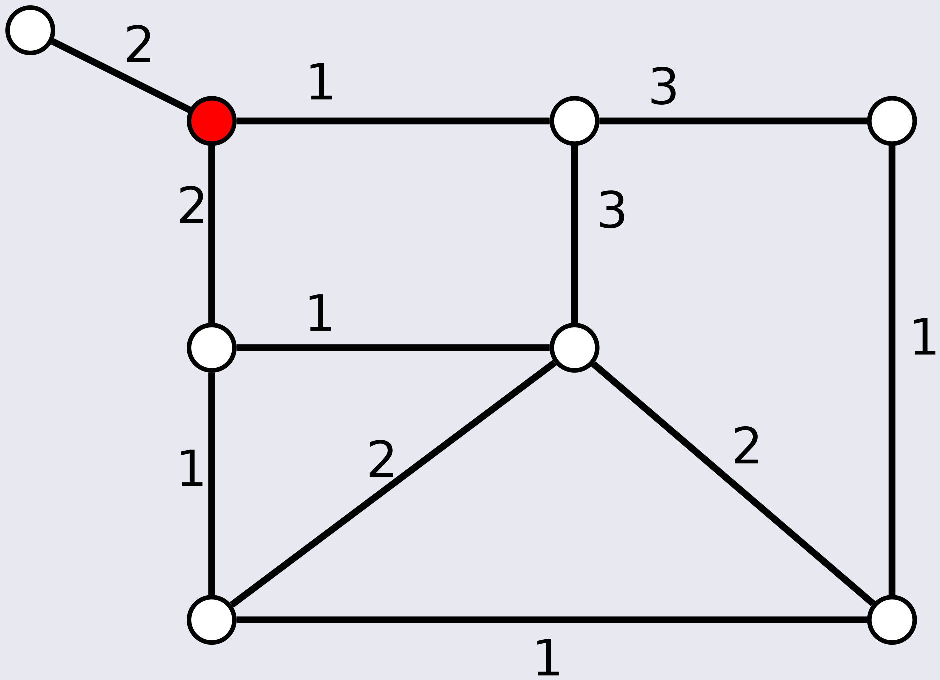
Objective:

- Find an ordering of the vertices $v_1, v_2, \ldots, v_n$ such that $d(v_1, v_2) + d(v_2, v_3) + \ldots + d(v_n, v_1)$ is minimized.
- $d(v_i, v_j)$ is the shortest-path distance of v_i, v_j on G

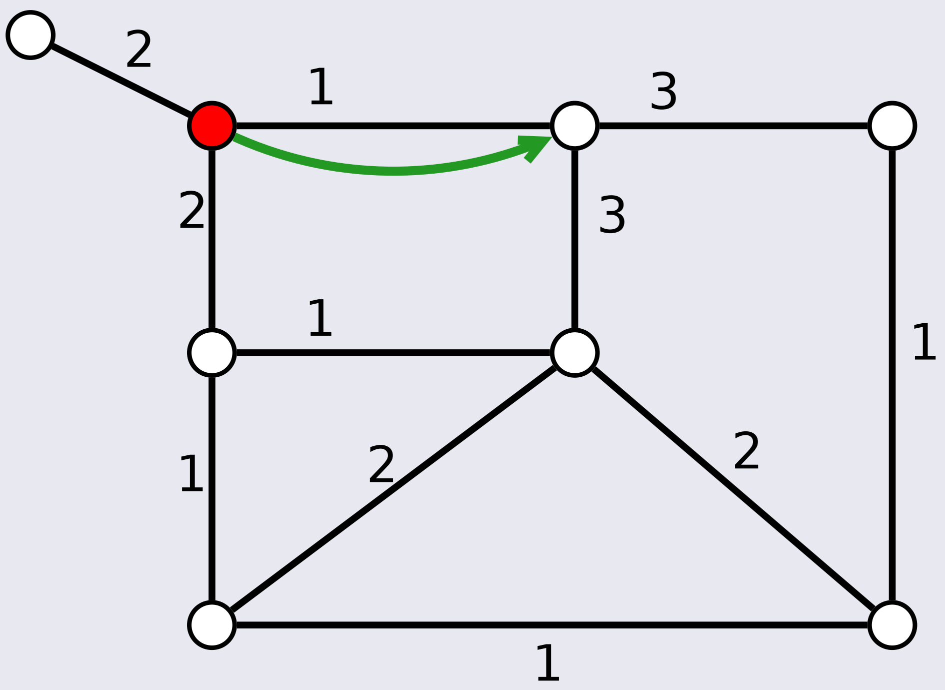
The Traveling Salesman Problem



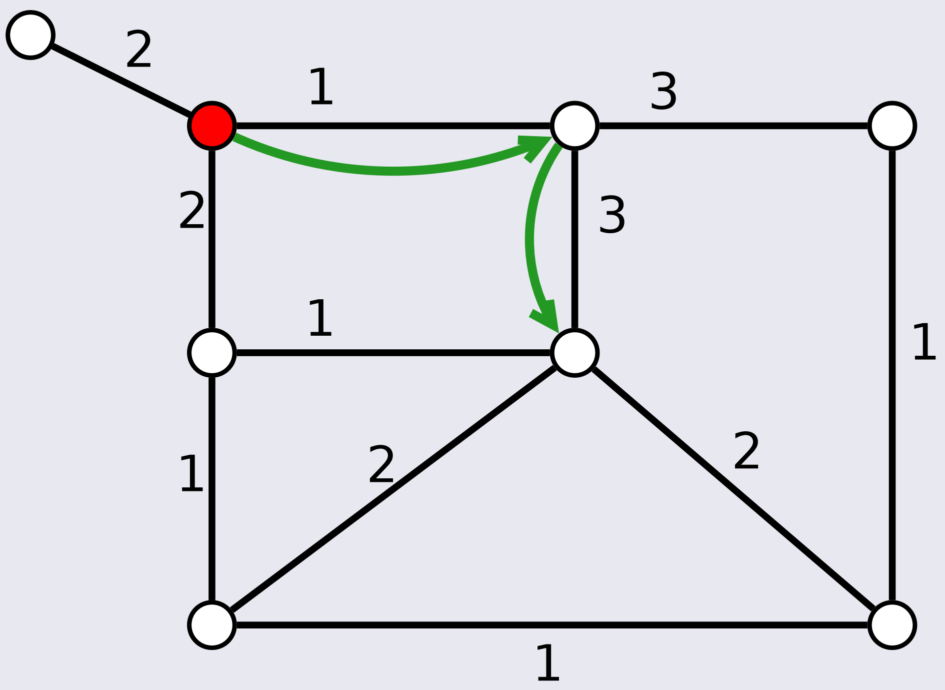
The Traveling Salesman Problem



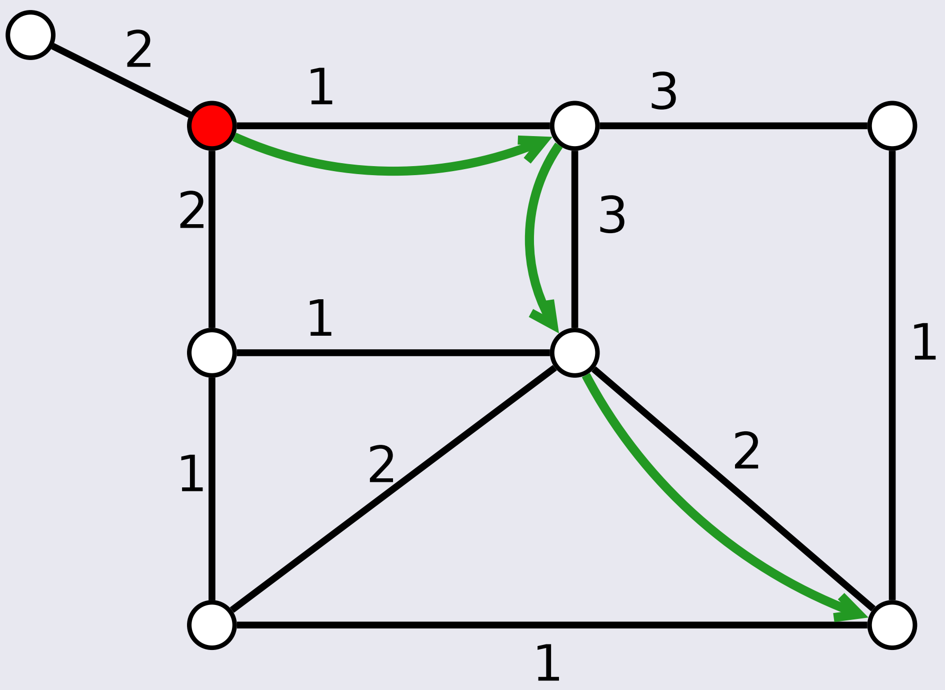
The Traveling Salesman Problem



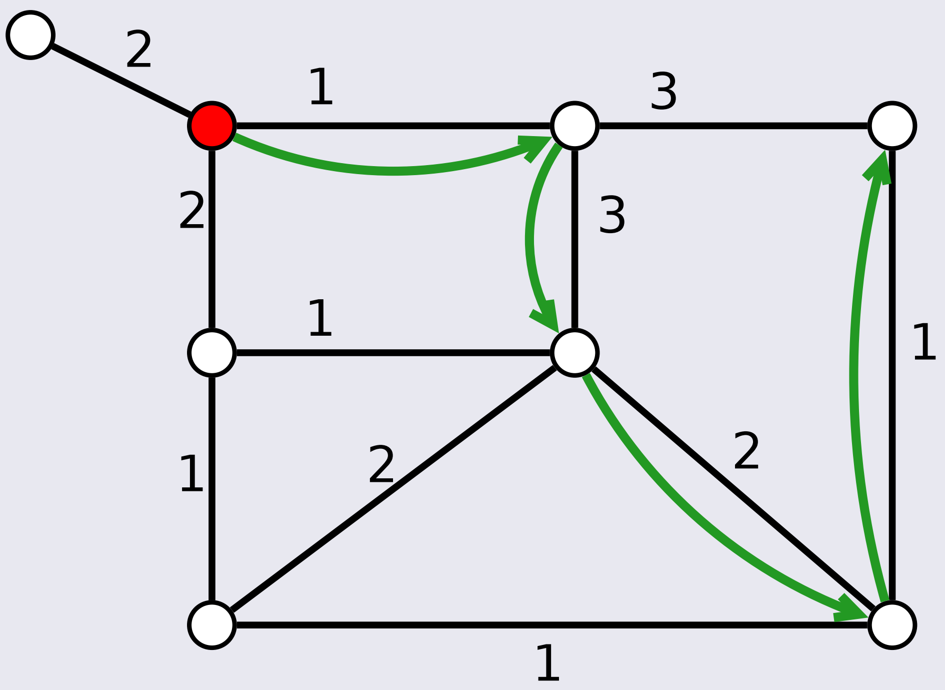
The Traveling Salesman Problem



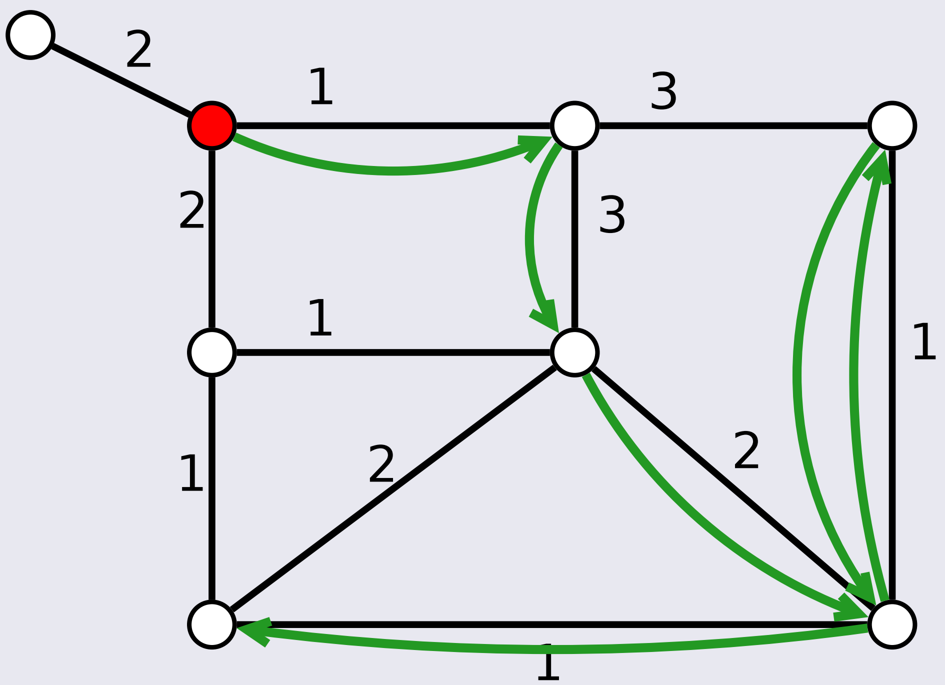
The Traveling Salesman Problem



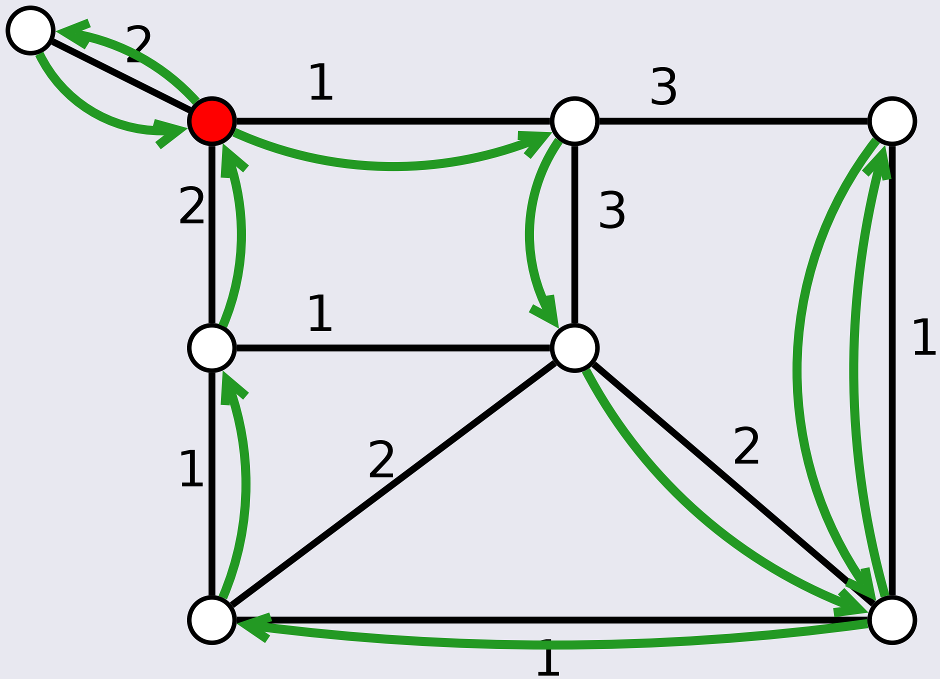
The Traveling Salesman Problem



The Traveling Salesman Problem



The Traveling Salesman Problem



TSP Approximations – Upper bounds

- $\frac{3}{2}$ approximation (Christofides 1976)

For graphic (un-weighted) case

- $\frac{3}{2} - \epsilon$ approximation (Oveis Gharan et al. FOCS '11)
- 1.461 approximation (Mömke and Svensson FOCS '11)
- $\frac{13}{9}$ approximation (Mucha STACS '12)
- 1.4 approximation (Sebö and Vygen arXiv '12)
- For ATSP the best ratio is $O(\log n / \log\log n)$ (Asadpour et al. SODA '10)

TSP Approximations – Lower bounds

- Problem is APX-hard (Papadimitriou and Yannakakis '93)
- $\frac{5381}{5380}$-inapproximable, ATSP $\frac{2805}{2804}$ (Engebretsen STACS '99)
- $\frac{3813}{3812}$-inapproximable (Böckenhauer et al. STACS '00)
- $\frac{220}{219}$-inapproximable, ATSP $\frac{117}{116}$ (Papadimitriou and Vempala STOC '00, Combinatorica '06)

TSP Approximations – Lower bounds

- Problem is APX-hard (Papadimitriou and Yannakakis '93)
- $\frac{5381}{5380}$-inapproximable, ATSP $\frac{2805}{2804}$ (Engebretsen STACS '99)
- $\frac{3813}{3812}$-inapproximable (Böckenhauer et al. STACS '00)
- $\frac{220}{219}$-inapproximable, ATSP $\frac{117}{116}$ (Papadimitriou and Vempala STOC '00, Combinatorica '06)

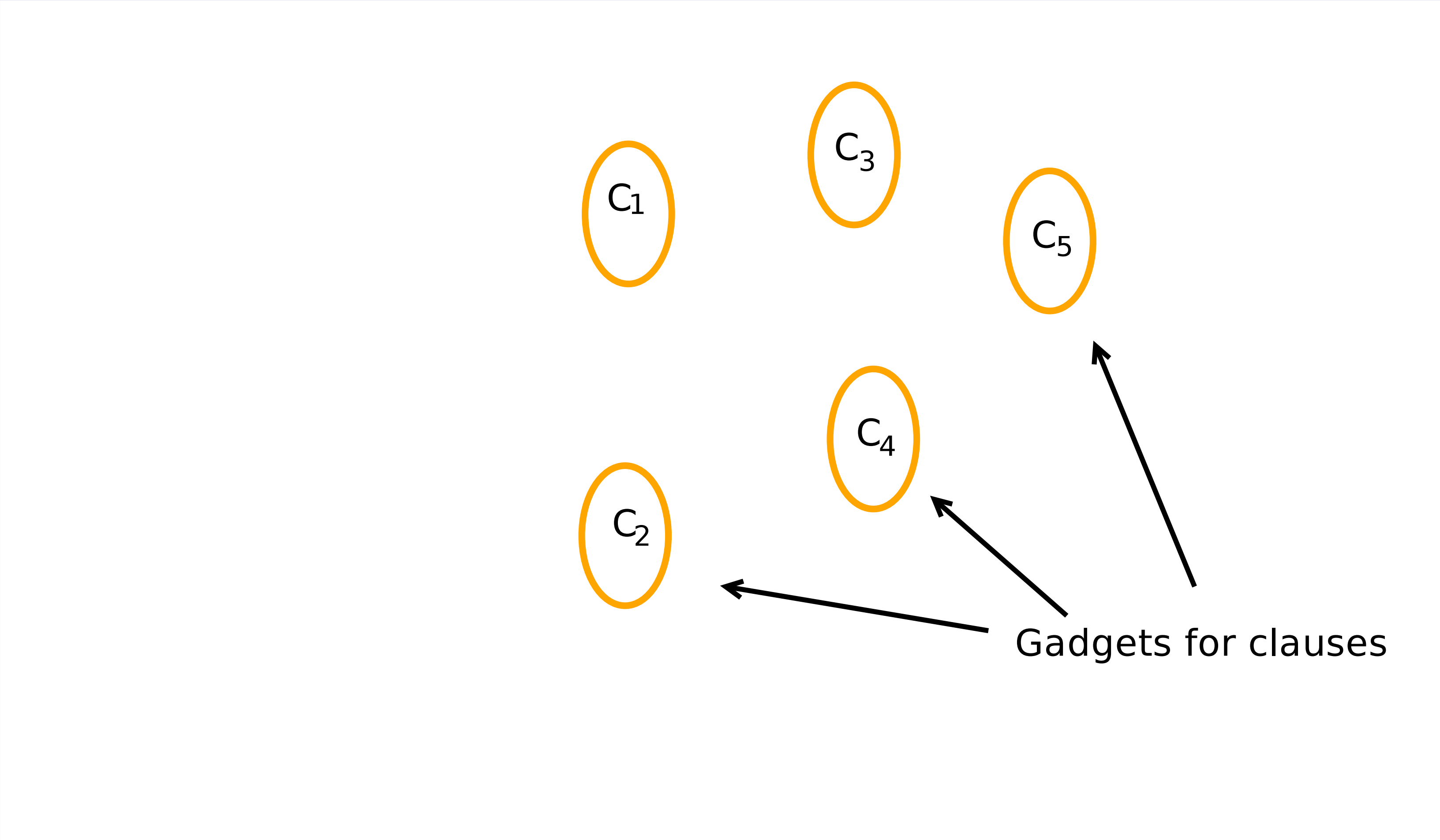
This talk:

Theorem
It is NP-hard to approximate TSP better than $\frac{123}{122}$ and ATSP better than $\frac{75}{74}$.

Reduction Technique

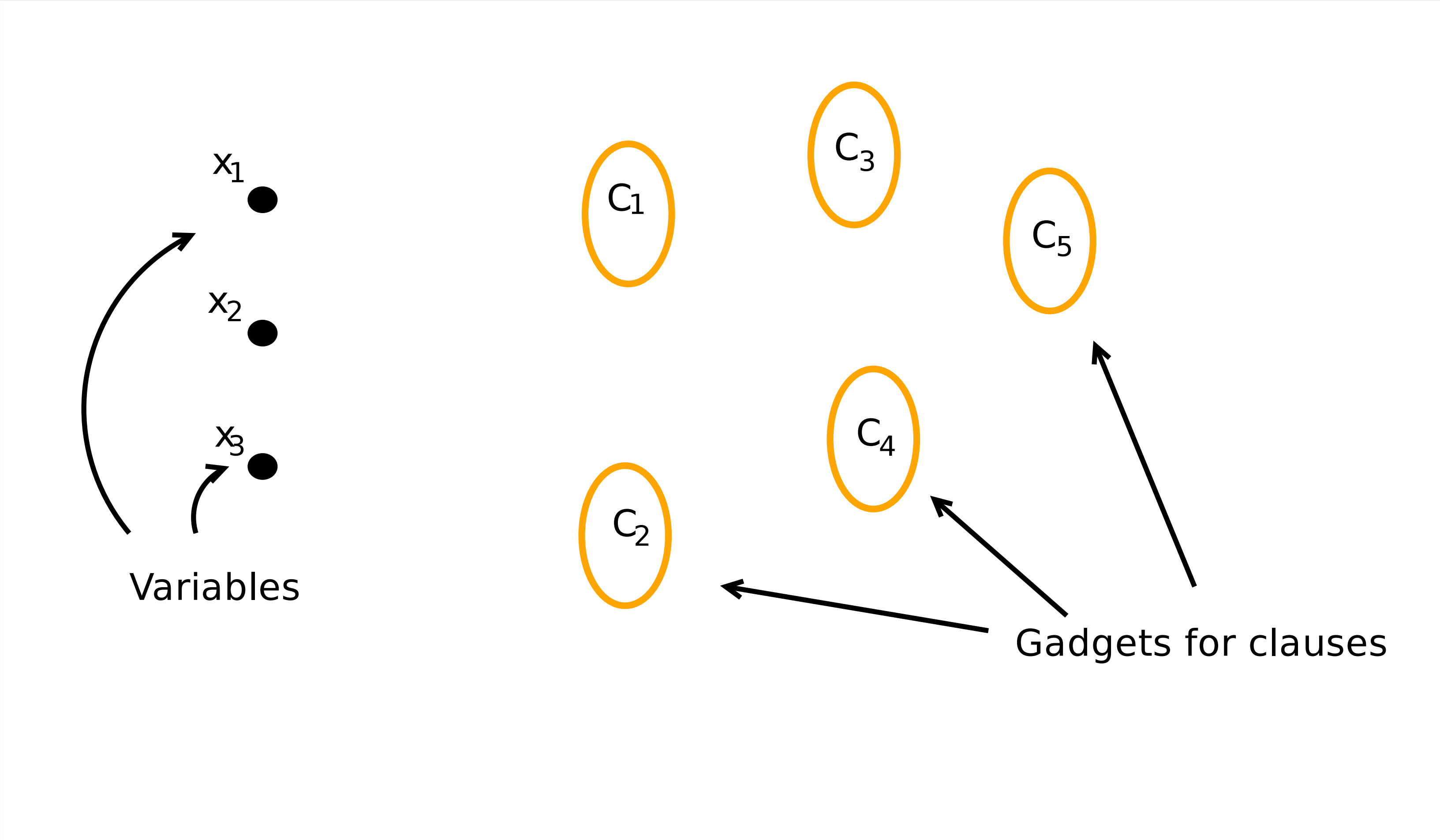
We reduce some inapproximable CSP (e.g. MAX-3SAT) to TSP.

Reduction Technique



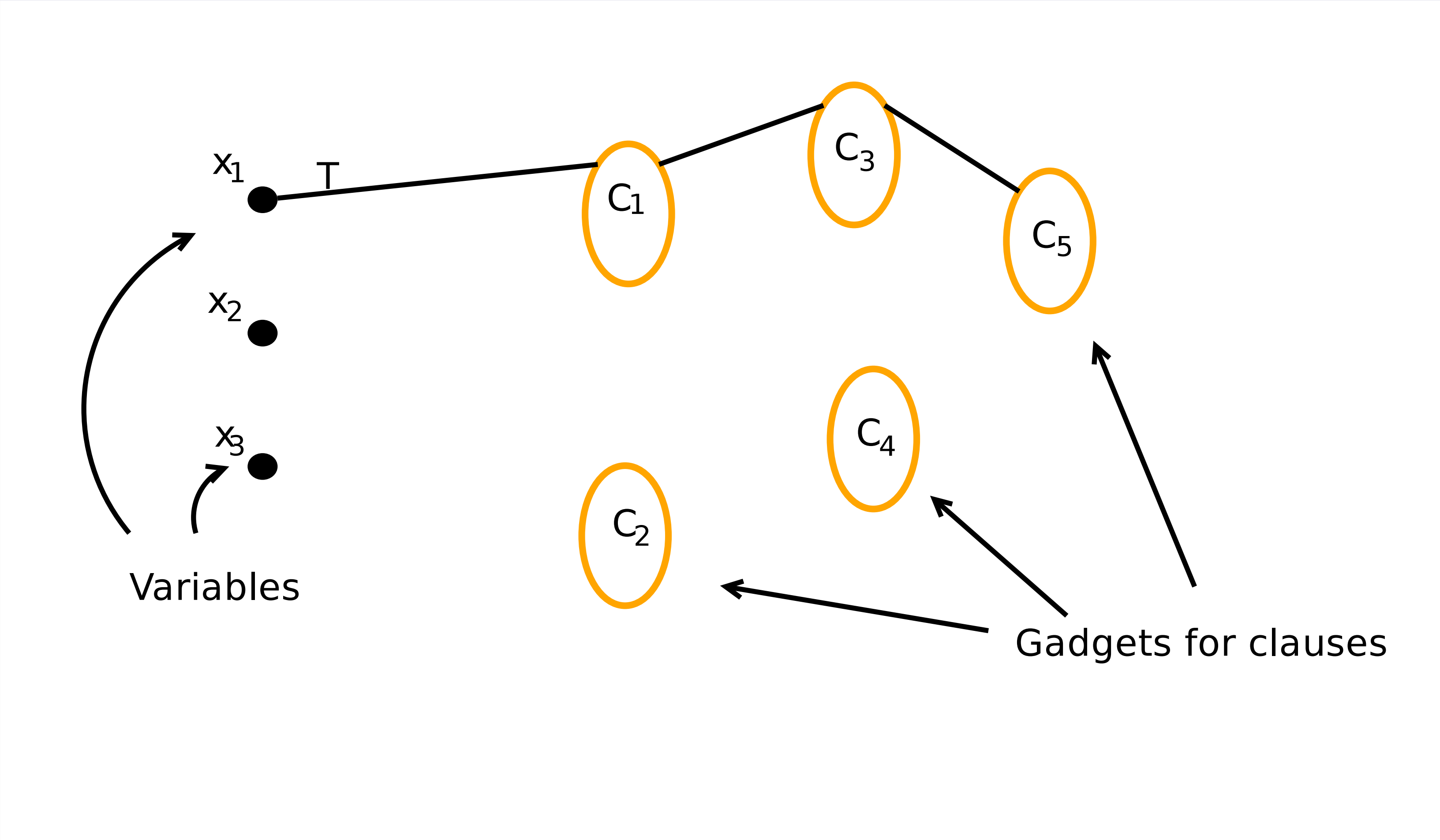
First, design some gadgets to represent the clauses

Reduction Technique



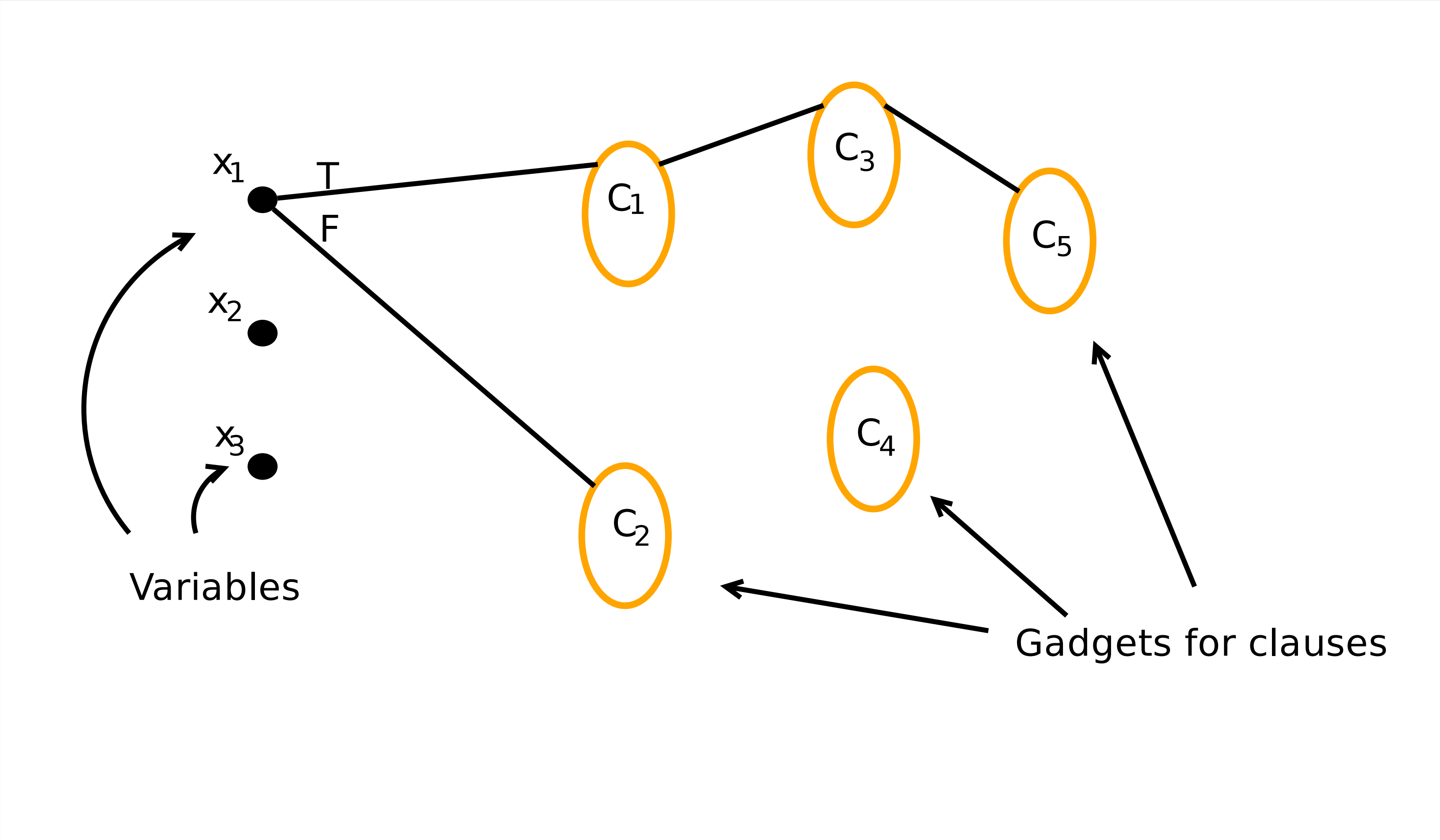
Then, add some choice vertices to represent truth assignments to variables

Reduction Technique



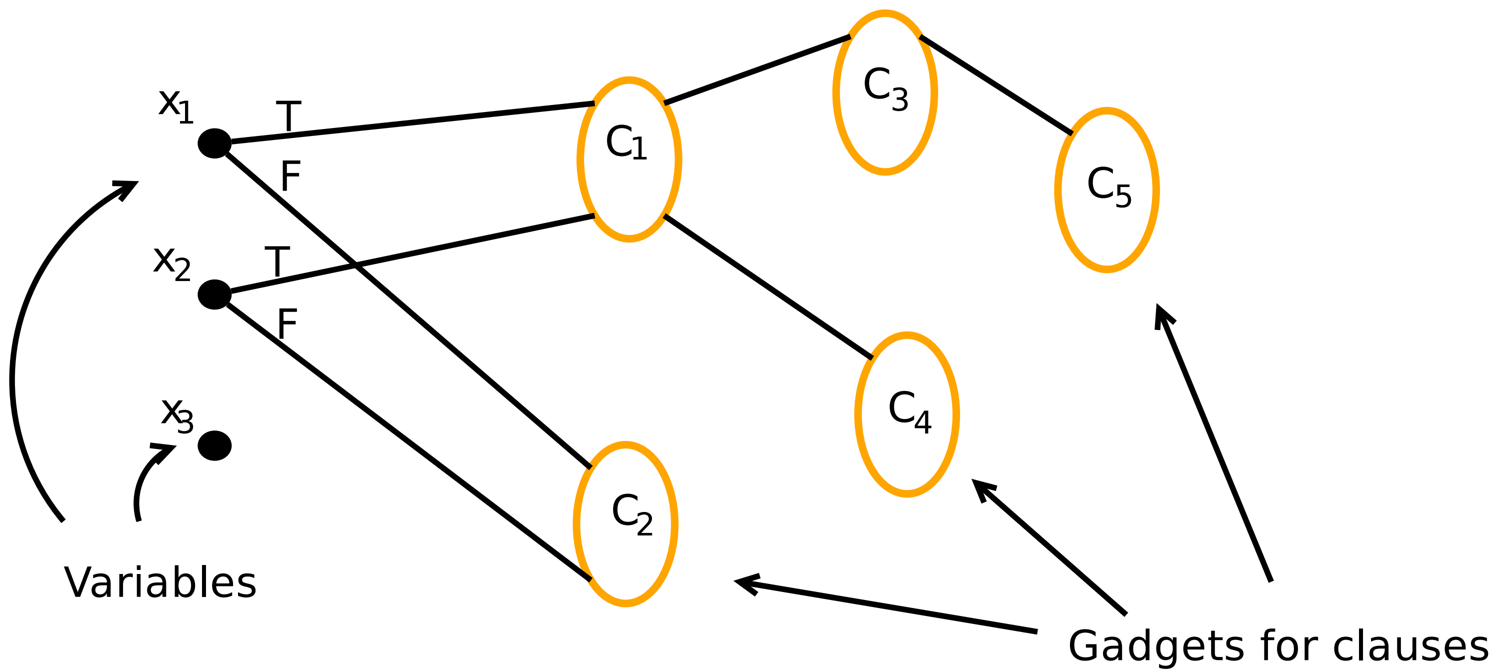
For each variable, create a path through clauses where it appears positive

Reduction Technique

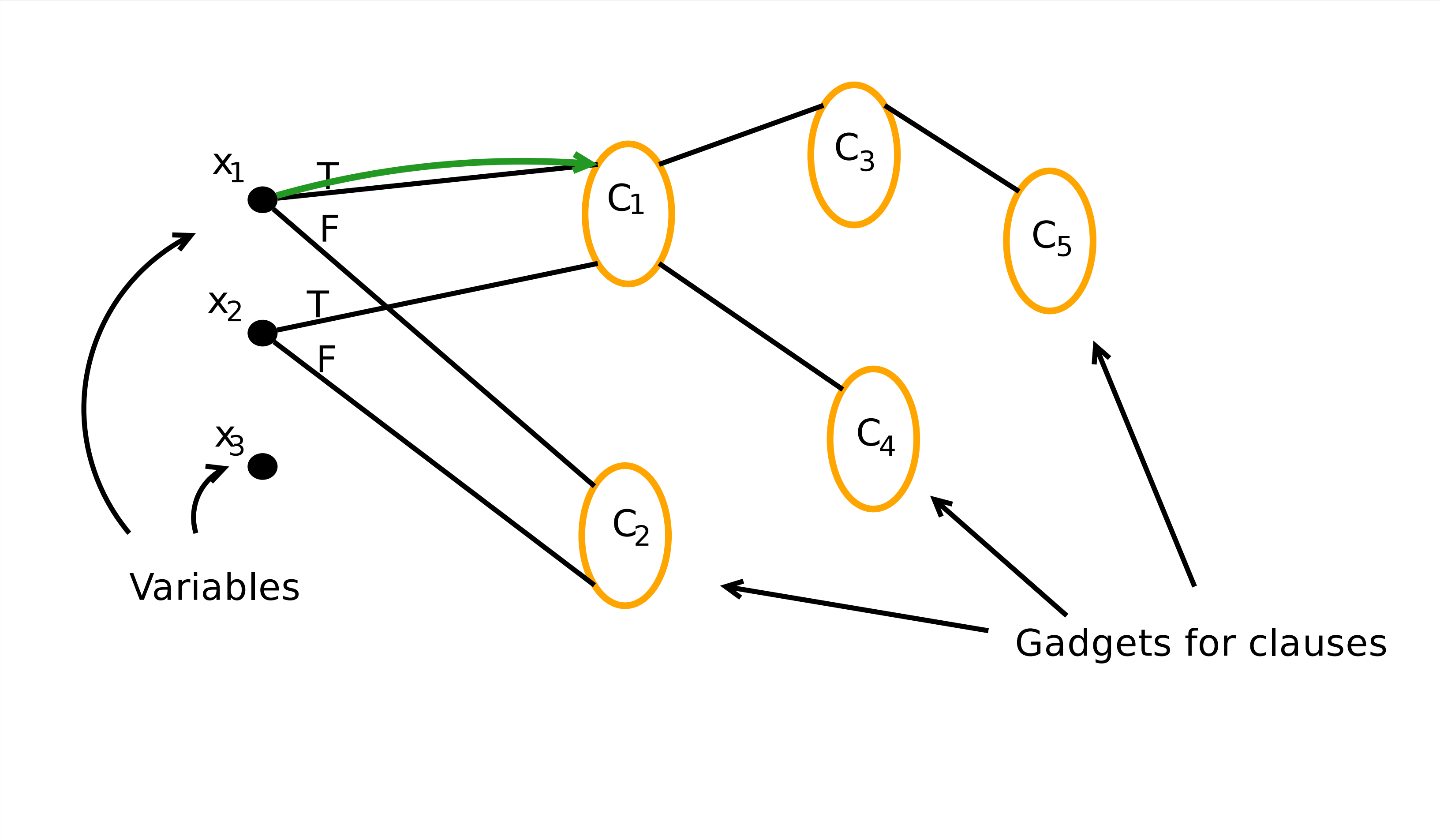


...and another path for its negative appearances

Reduction Technique

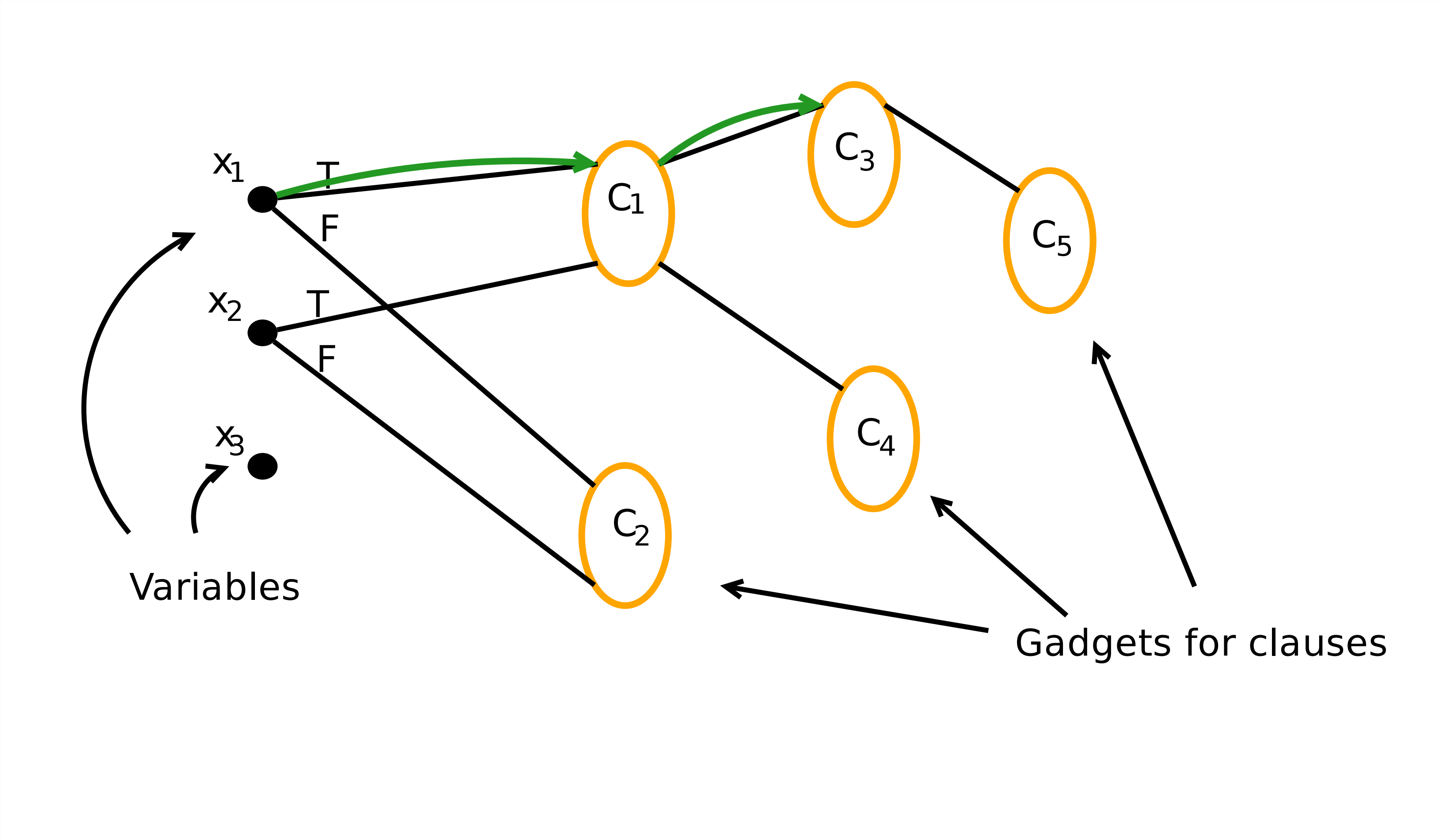


Reduction Technique

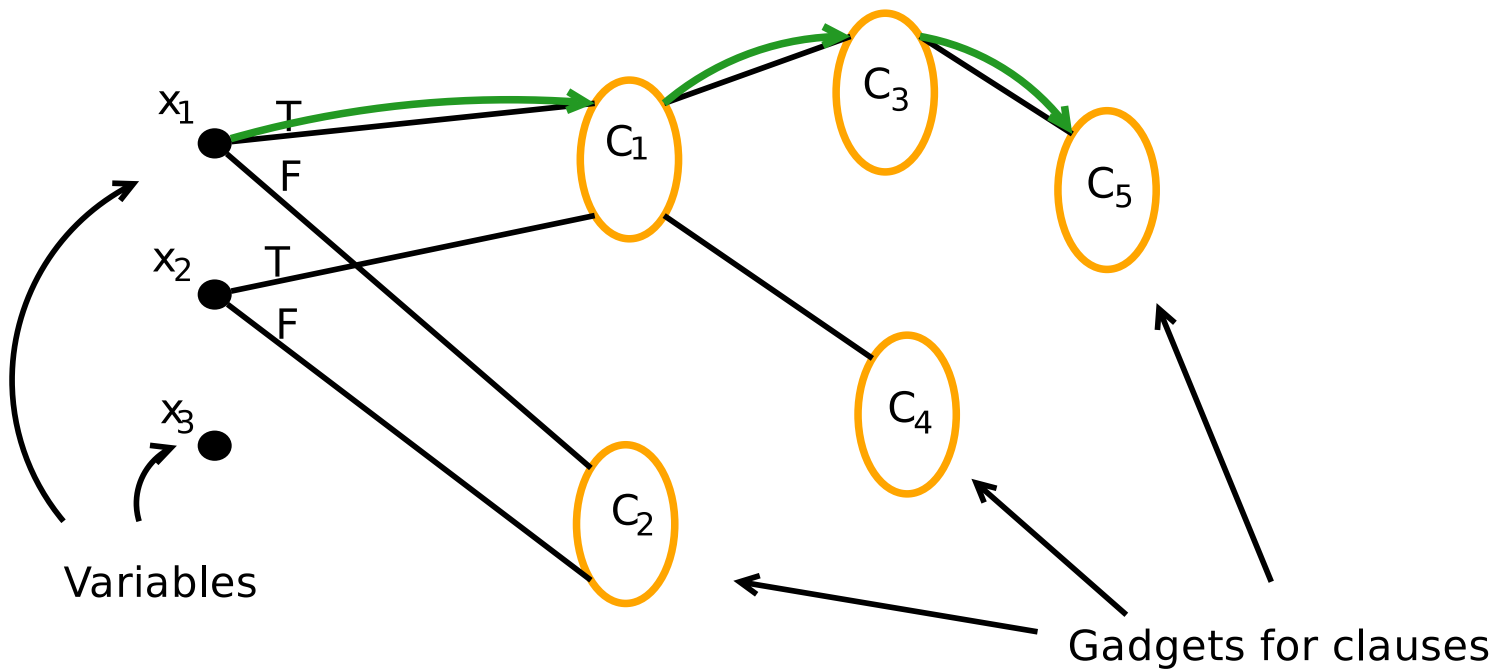


A truth assignment dictates a general path

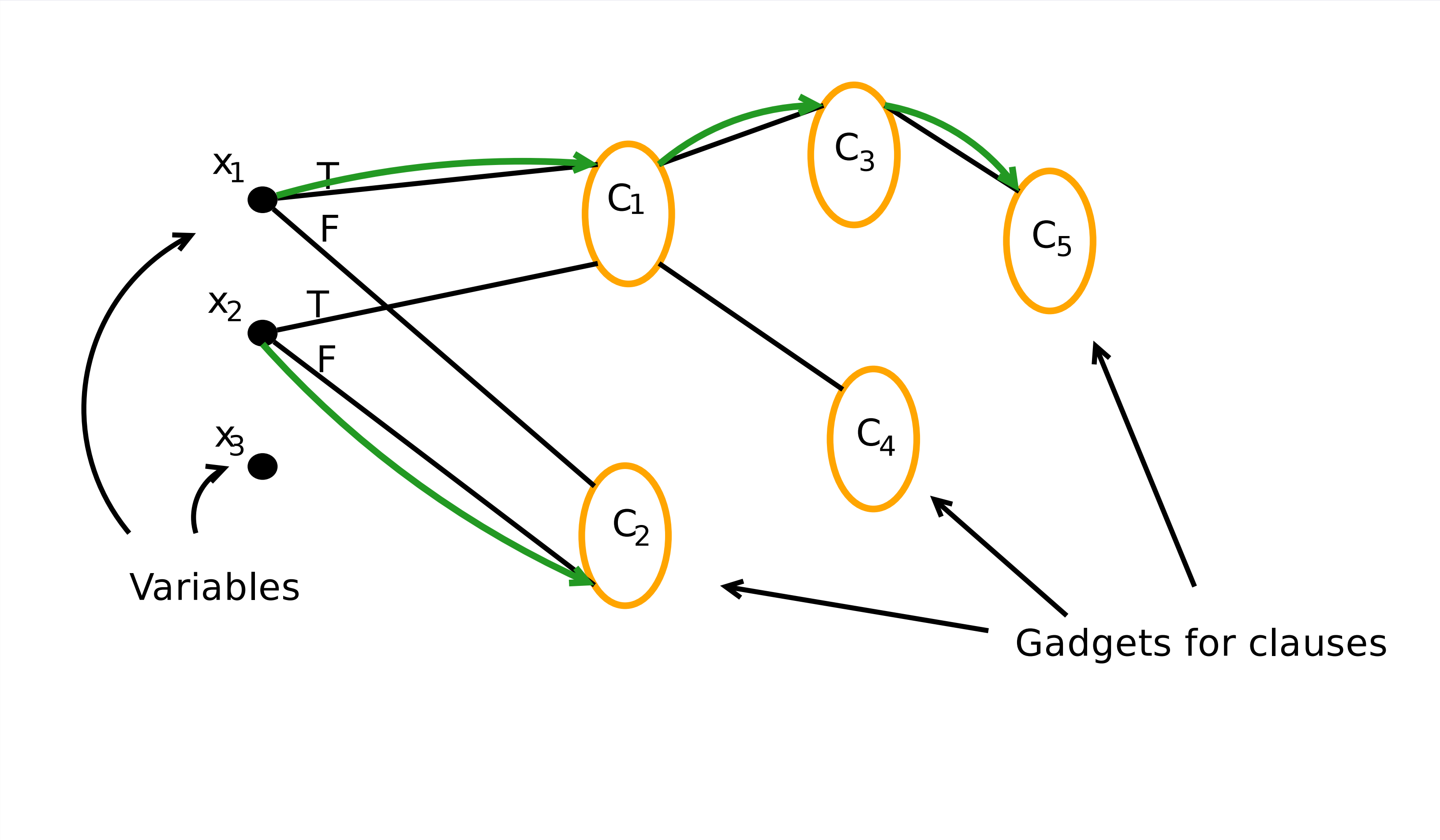
Reduction Technique



Reduction Technique

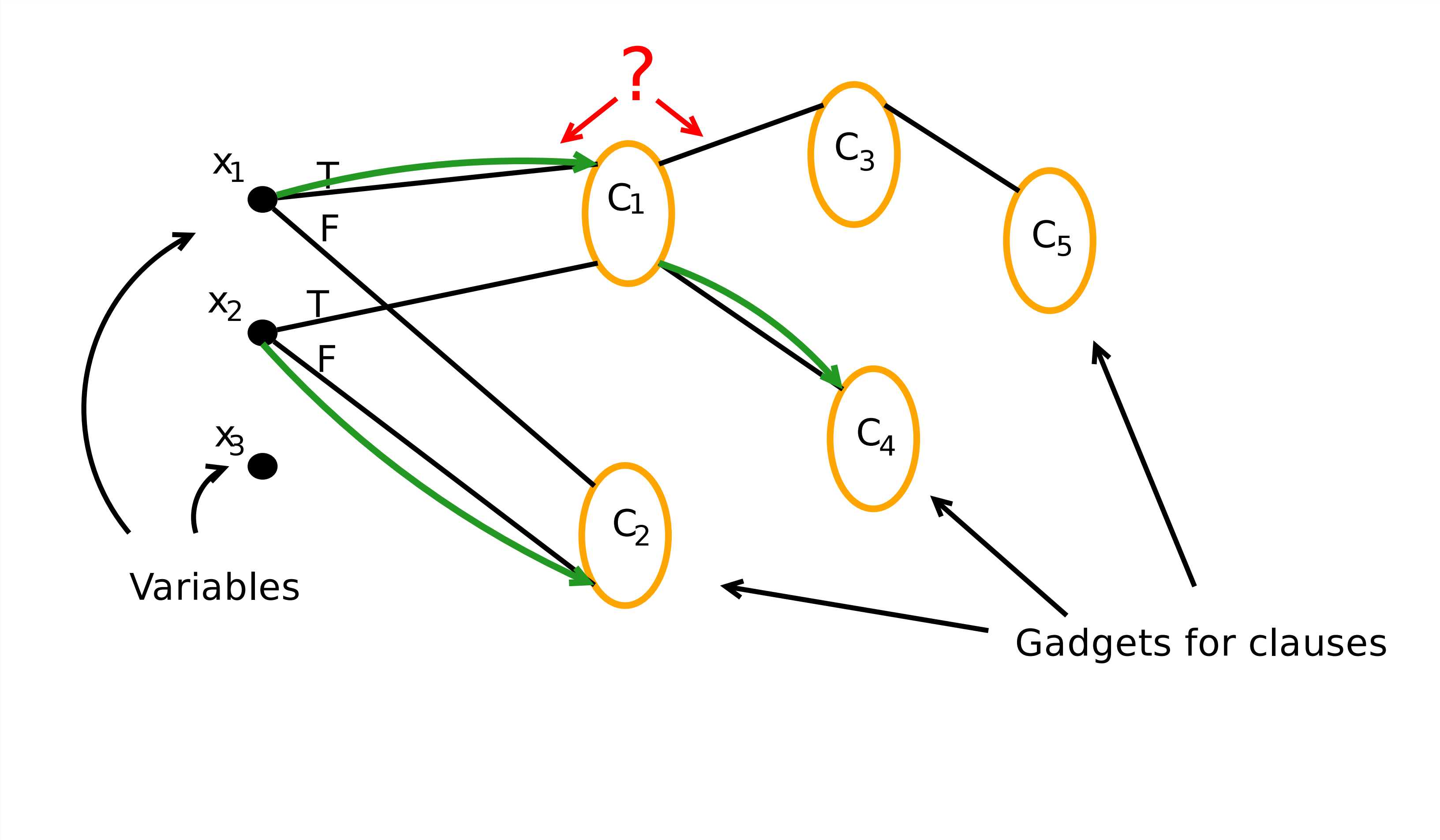


Reduction Technique



We must make sure that gadgets are cheaper to traverse if corresponding clause is satisfied

Reduction Technique



For the converse direction we must make sure that ”cheating” tours are not optimal!

How to ensure consistency

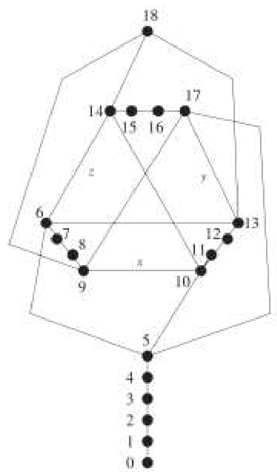


Figure 6. Equation gadget for the symmetric TSP

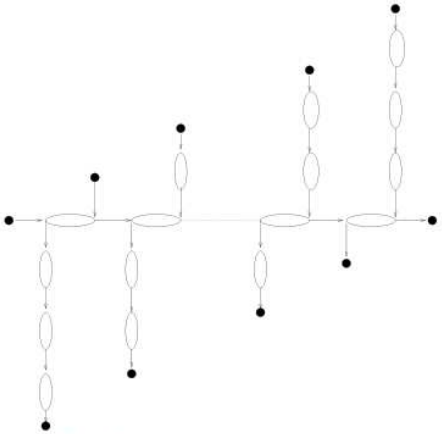


Figure 3. The neighborhood of an edge gadget

- Papadimitriou and Vempala design a gadget for Parity.
- They eliminate variable vertices altogether.
- Consistency is achieved by hooking up gadgets ”randomly”
  - In fact gadgets that share a variable are connected according to the structure dictated by a special graph
  - The graph is called a ”pusher”. Its existence is proved using the probabilistic method.

How to ensure consistency

- Basic idea here: consistency would be easy if each variable occurred at most c times, c a constant.
  - Cheating would only help a tour ”fix” a bounded number of clauses.

How to ensure consistency

- Basic idea here: consistency would be easy if each variable occurred at most c times, c a constant.
  - Cheating would only help a tour ”fix” a bounded number of clauses.
- We will rely on techniques and tools used to prove inapproximability for bounded-occurrence CSPs.
  - This is where expander graphs are important.
  - Main tool: “amplifier graph” constructions due to Berman and Karpinski.

How to ensure consistency

- Basic idea here: consistency would be easy if each variable occurred at most c times, c a constant.
  - Cheating would only help a tour "fix" a bounded number of clauses.
- We will rely on techniques and tools used to prove inapproximability for bounded-occurrence CSPs.
  - This is where expander graphs are important.
  - Main tool: "amplifier graph" constructions due to Berman and Karpinski.
- Result: an easier hardness proof that can be broken down into independent pieces, and also gives improved bounds.

Expander and Amplifier Graphs

Expander Graphs

- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

Expander Graphs

- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

- Definition:

  A graph $G(V, E)$ is an expander if

  - For all $S \subseteq V$ with $|S| \leq \frac{|V|}{2}$ we have for some constant c

    $$\frac{|E(S, V \setminus S)|}{|S|} \geq c$$

  - The maximum degree Δ is bounded

Expander Graphs

- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Expander Graphs

- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

Expander Graphs

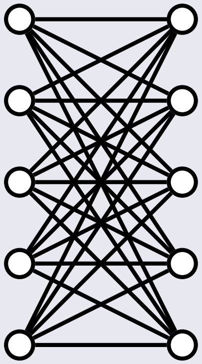
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

A complete bipartite graph is well-connected but not sparse.



Expander Graphs

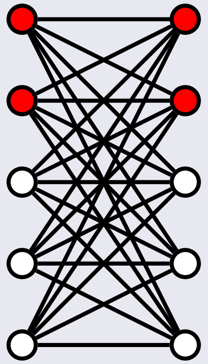
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

A complete bipartite graph is well-connected but not sparse.



Expander Graphs

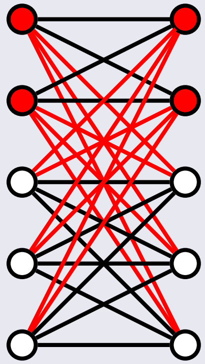
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

A complete bipartite graph is well-connected but not sparse.



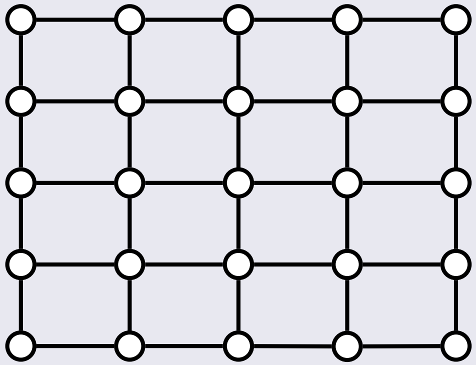
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

A grid is sparse but not well-connected.



Expander Graphs

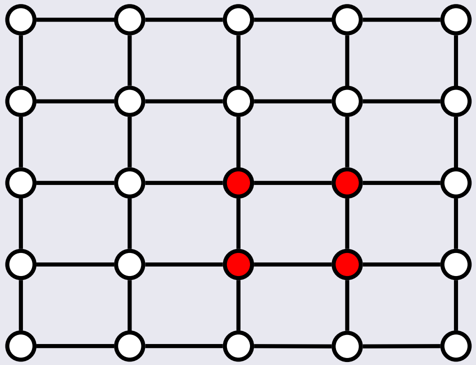
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

A grid is sparse but not well-connected.



Expander Graphs

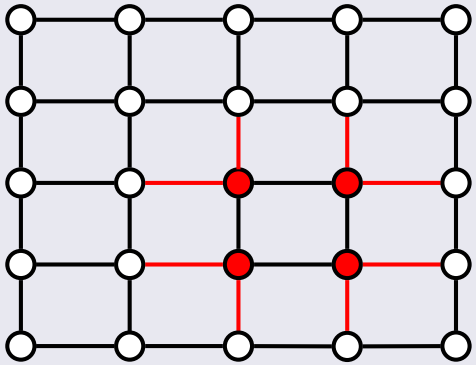
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

A grid is sparse but not well-connected.



Expander Graphs

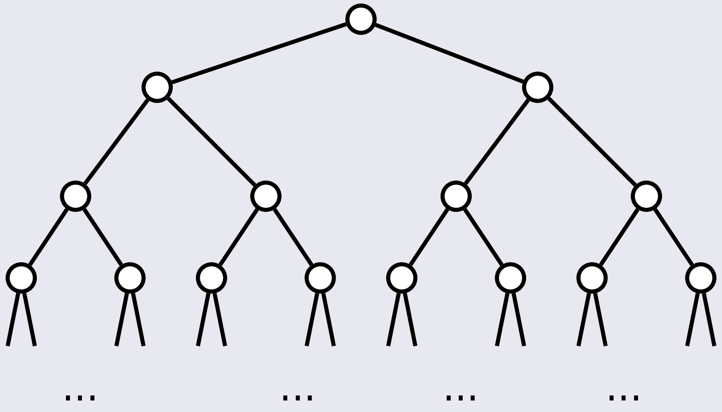
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

An infinite binary tree is a good expander.



Expander Graphs

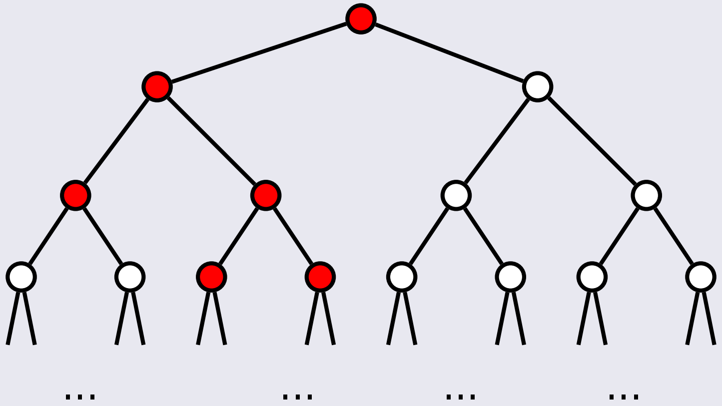
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

An infinite binary tree is a good expander.



Expander Graphs

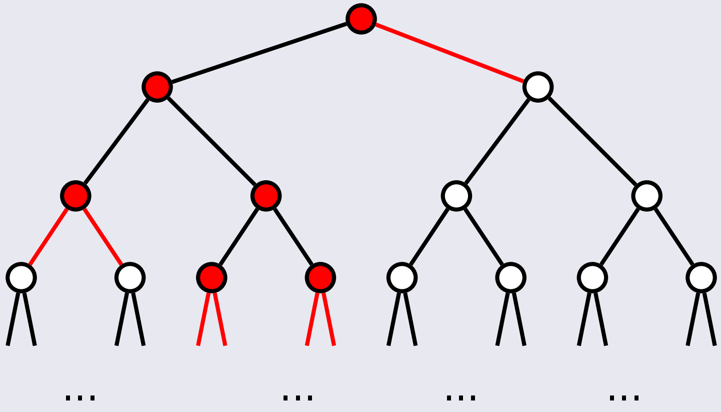
- Informal description:

  An expander graph is a **well-connected** and **sparse** graph.

  - In any possible partition of the vertices into two sets, there are **many** edges crossing the cut.
  - This is achieved even though the graph has low degree, therefore few edges.

Example:

An infinite binary tree is a good expander.



Applications of Expanders

Expander graphs have a number of applications

- Proof of PCP theorem
- Derandomization
- Error-correcting codes

Applications of Expanders

Expander graphs have a number of applications

- Proof of PCP theorem
- Derandomization
- Error-correcting codes
- ... and inapproximability of bounded occurrence CSPs!

Expanders and inapproximability

- Consider the standard reduction from 3-SAT to 3-OCC-3-SAT
  - Replace each appearance of variable x with a fresh variable $x_1, x_2, \ldots, x_n$
  - Add the clauses $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \ldots \wedge (x_n \to x_1)$

Applications of Expanders

Expanders and inapproximability

- Consider the standard reduction from 3-SAT to 3-OCC-3-SAT
  - Replace each appearance of variable x with a fresh variable $x_1, x_2, \ldots, x_n$
  - Add the clauses $(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \ldots \wedge (x_n \to x_1)$

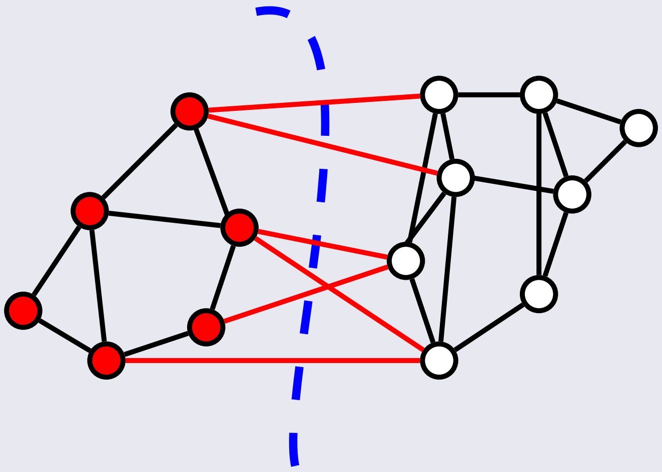
Problem: This does not preserve inapproximability!

  - We could add $(x_i \to x_j)$ for all i, j.
  - This ensures consistency but adds too many clauses and does not decrease number of occurrences!

Applications of Expanders

Expanders and inapproximability

- We modify this using a 1-expander [Papadimitriou Yannakakis 91]
  - Recall: a 1-expander is a graph s.t. in each partition of the vertices the number of edges crossing the cut is larger than the number of vertices of the smaller part.



Expanders and inapproximability

- We modify this using a 1-expander [Papadimitriou Yannakakis 91]
  - Replace each appearance of variable x with a fresh variable $x_1, x_2, \dots, x_n$
  - Construct an n-vertex 1-expander.
  - For each edge (i, j) add the clauses $(x_i \to x_j) \wedge (x_j \to x_i)$

Why does this work?

- Suppose that in the new instance the optimal assignment sets some of the x_i's to 0 and others to 1.
- This gives a partition of the 1-expander.
- Each edge cut by the partition corresponds to an unsatisfied clause.
- Number of cut edges $>$ number of minority assigned vertices $=$ number of clauses lost by being consistent.

Hence, it is always optimal to give the same value to all x_i's.

- Also, because expander graphs are sparse, only linear number of clauses added.
- This gives some inapproximability constant.

Limits of expanders

- Expanders sound useful. But how good expanders can we get?

  We want:

  - Low degree – few edges
  - High expansion (at least 1).

  These are conflicting goals!

Limits of expanders

- Expanders sound useful. But how good expanders can we get?

  We want:

  - Low degree – few edges
  - High expansion (at least 1).

  These are conflicting goals!

- The smallest Δ for which we currently know we can have expansion 1 is $\Delta = 6$. [Bollobás 88]

Limits of expanders

- Expanders sound useful. But how good expanders can we get?

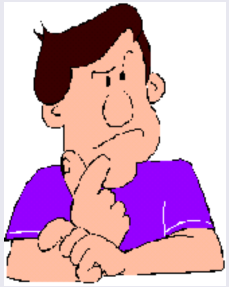
  We want:

  - Low degree – few edges
  - High expansion (at least 1).

  These are conflicting goals!

- The smallest Δ for which we currently know we can have expansion 1 is $\Delta = 6$. [Bollobás 88]

- Problem: $\Delta = 6$ is too large, $\Delta = 5$ probably won't work...



Amplifiers

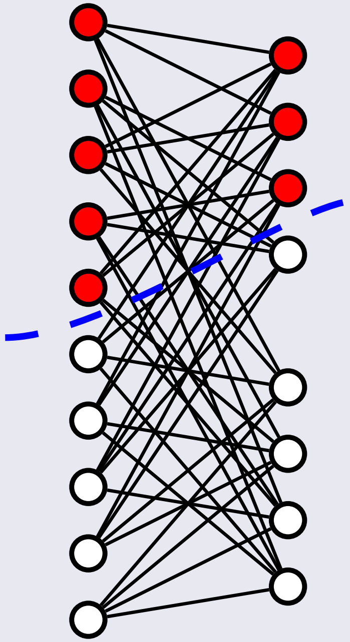
- Amplifiers are expanders for some of the vertices.
- The other vertices are thrown in to make consistency easier to achieve.
- This allows us to get smaller Δ.

Amplifiers

- Amplifiers are expanders for some of the vertices.
- The other vertices are thrown in to make consistency easier to achieve.
- This allows us to get smaller Δ.

5-regular amplifier [Berman Karpinski 03]

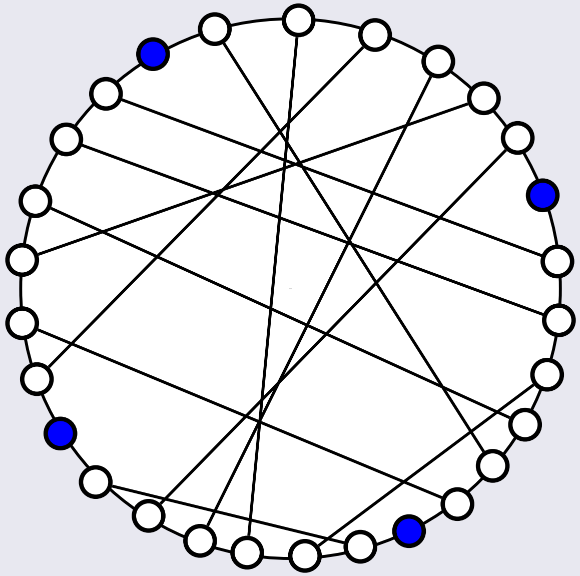
- Bipartite graph. n vertices on left, $0.8n$ vertices on right.
- 4-regular on left, 5-regular on right.
- Graph constructed randomly.
- Crucial Property: whp any partition cuts more edges than the number of left vertices on the smaller set.



Amplifiers

- Amplifiers are expanders for some of the vertices.
- The other vertices are thrown in to make consistency easier to achieve.
- This allows us to get smaller Δ.

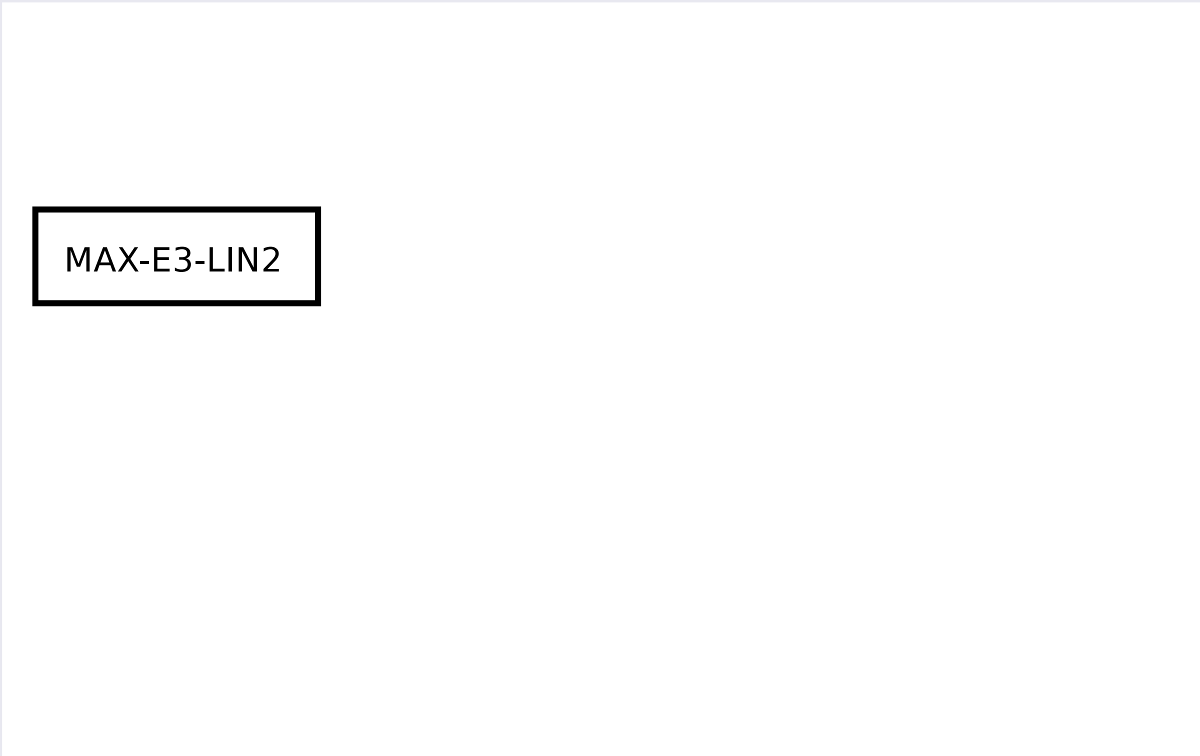
3-regular wheel amplifier [Berman Karpinski 01]



- Start with a cycle on $7n$ vertices.
- Every seventh vertex is a contact vertex. Other vertices are checkers.
- Take a random perfect matching of checkers.

Back to the Reduction

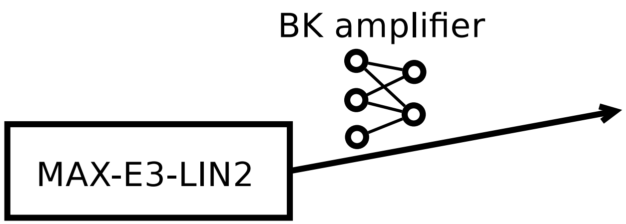
Overview



We start from an instance of MAX-E3-LIN2. Given a set of linear equations (mod 2) each of size three satisfy as many as possible.

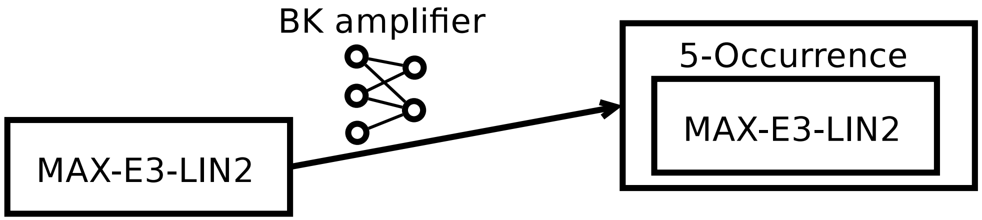
Problem known to be 2-inapproximable (Håstad)

Overview

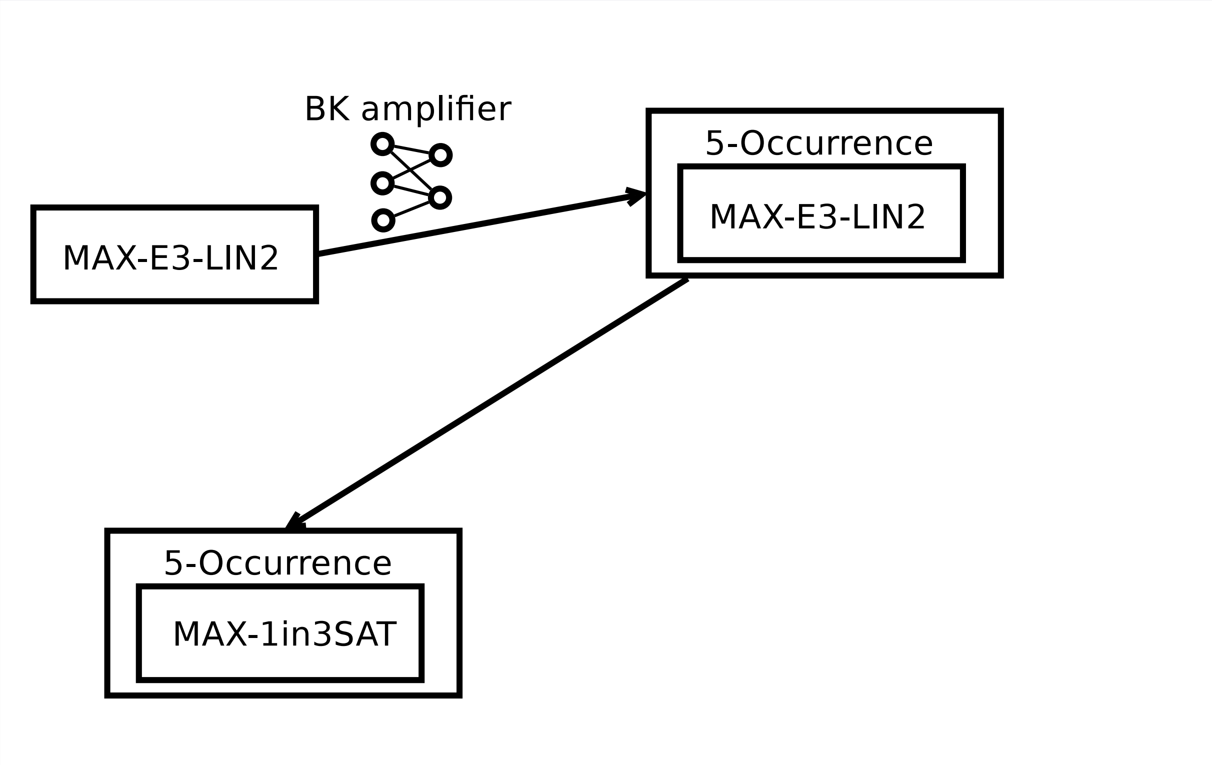


We use the Berman-Karpinski amplifier construction to obtain an instance where each variable appears exactly 5 times (and most equations have size 2).

Overview

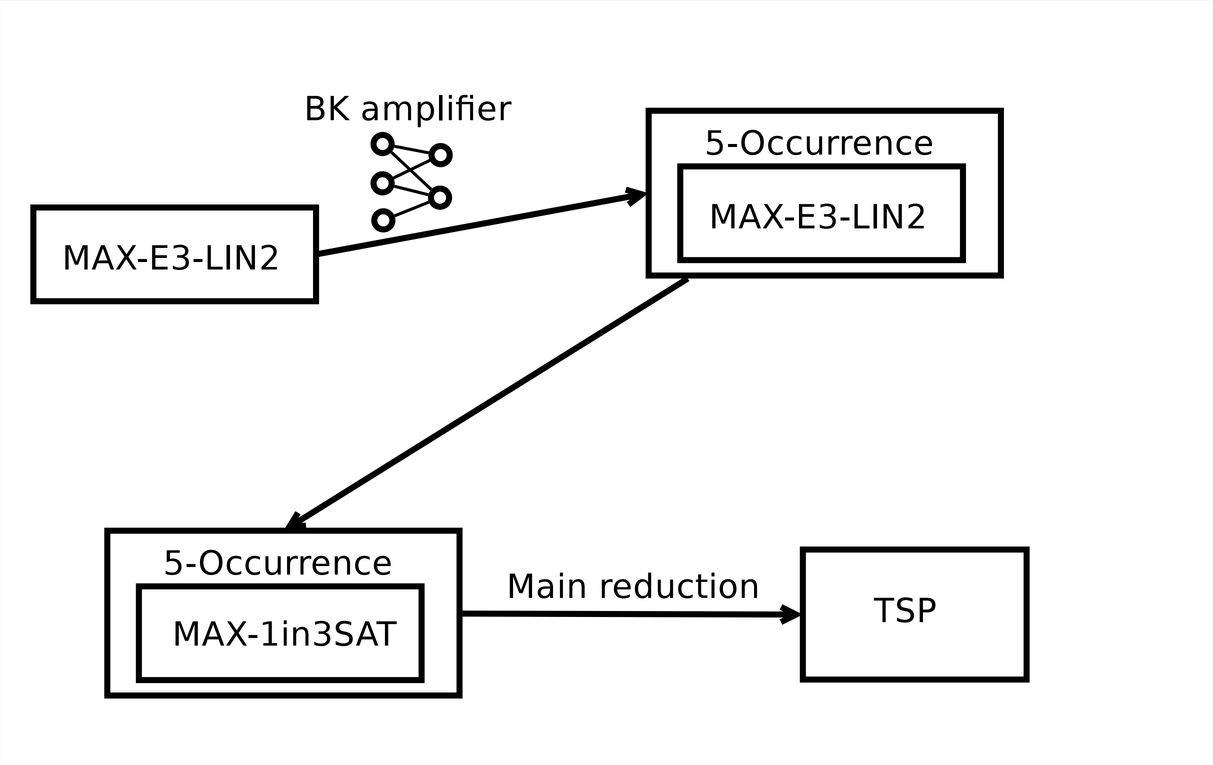


Overview



A simple trick reduces this to the 1in3 predicate.

Overview



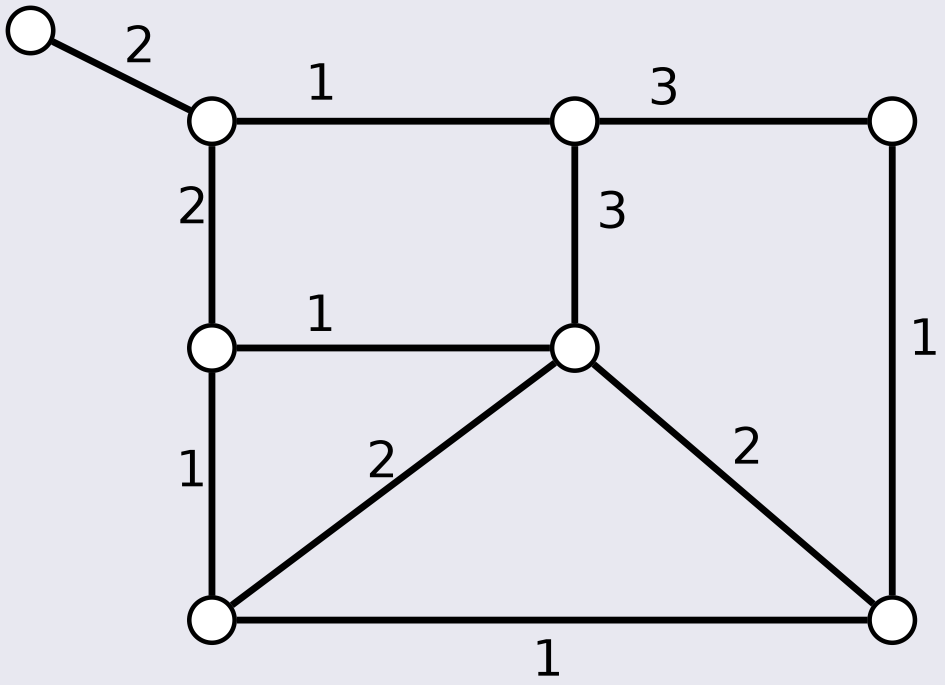
From this instance we construct a graph.

1in3-SAT

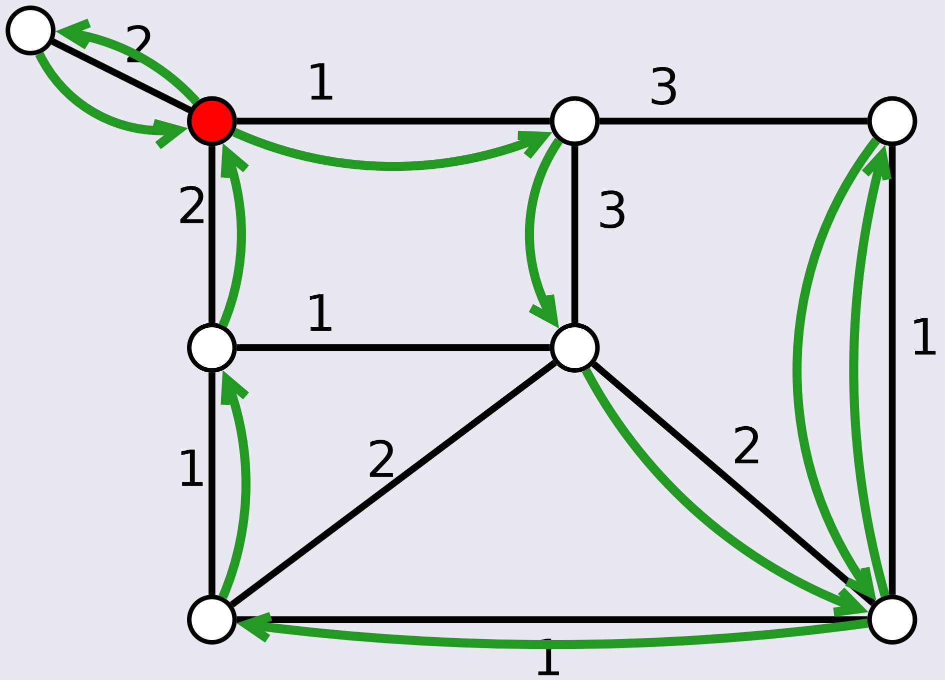
Input:
A set of clauses $(l_1 \vee l_2 \vee l_3)$, l_1, l_2, l_3 literals.
Objective:
A clause is satisfied if exactly one of its literals is true. Satisfy as many clauses as possible.

- Easy to reduce MAX-LIN2 to this problem.
  - Especially for size two equations $(x + y = 1) \leftrightarrow (x \vee y)$.
- Naturally gives gadget for TSP
  - In TSP we'd like to visit each vertex at least once, but not more than once (to save cost)

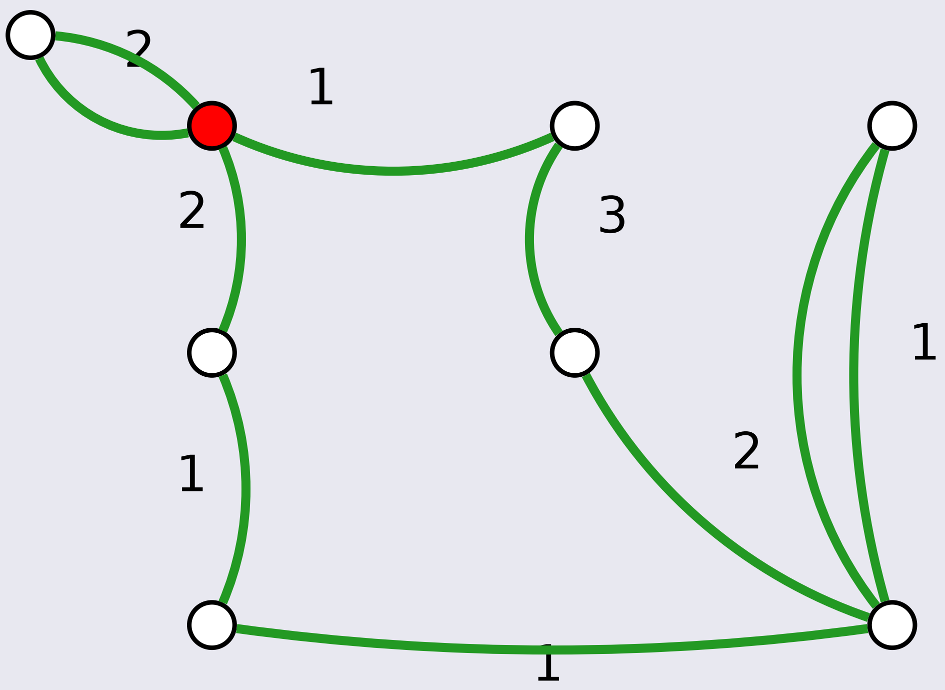
TSP and Euler tours



TSP and Euler tours



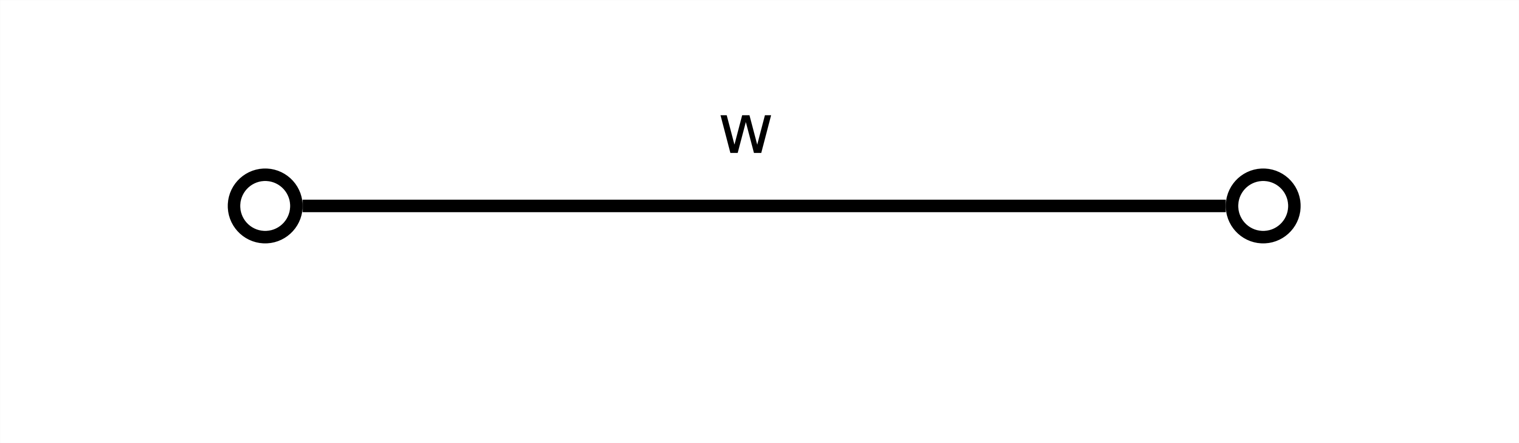
TSP and Euler tours



TSP and Euler tours

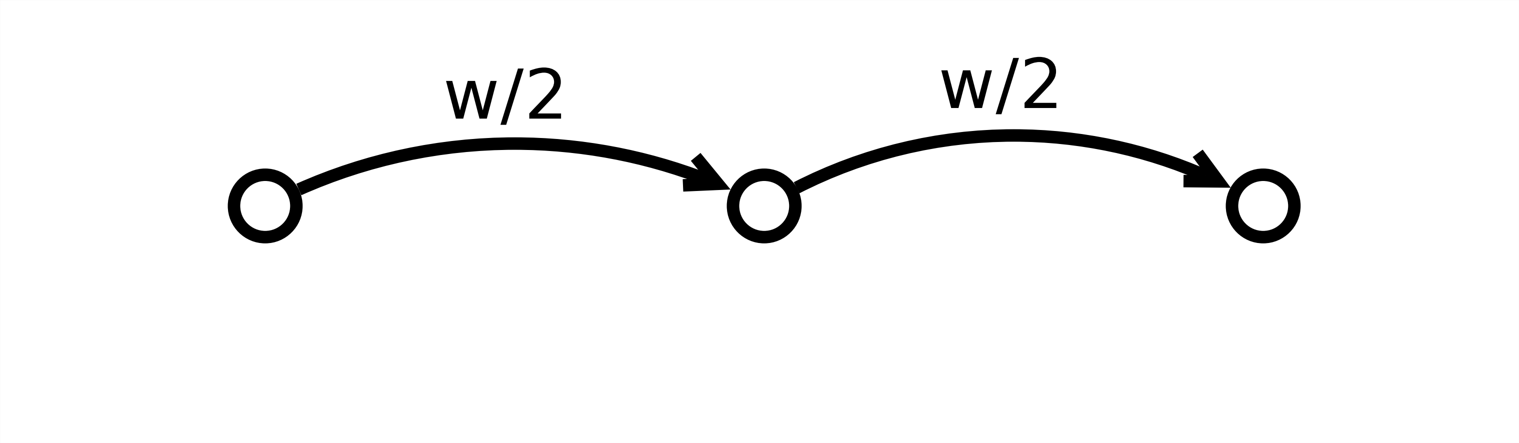
- A TSP tour gives an Eulerian multi-graph composed with edges of G.
- An Eulerian multi-graph composed with edges of G gives a TSP tour.
  - TSP $\equiv$ Select a multiplicity for each edge so that the resulting multi-graph is Eulerian and total cost is minimized
  - **Note**: no edge is used more than twice

Gadget – Forced Edges



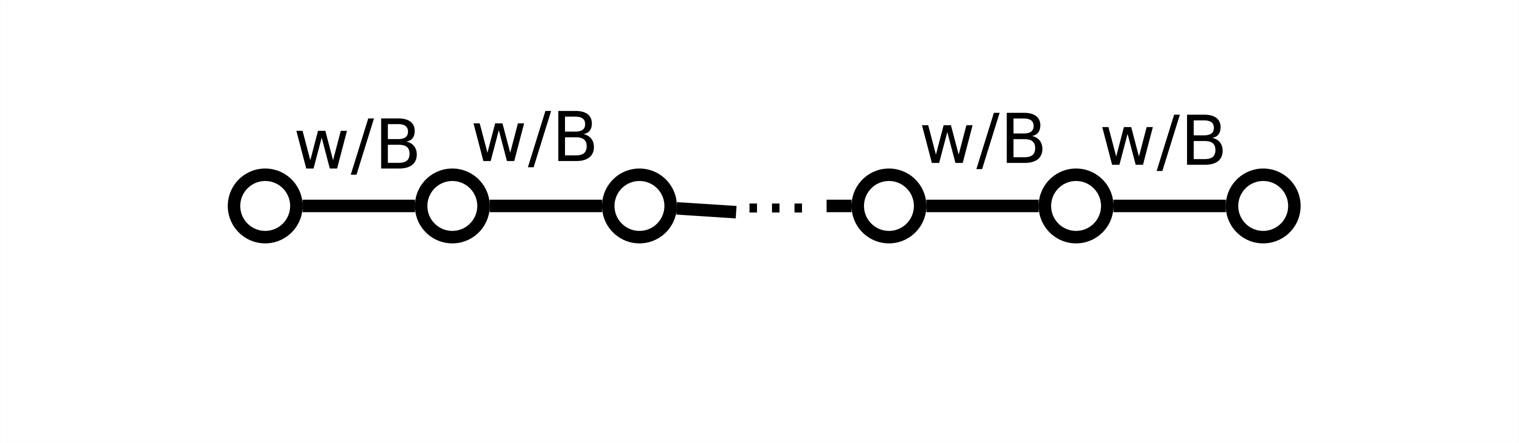
We would like to be able to dictate in our construction that a certain edge has to be used at least once.

Gadget – Forced Edges



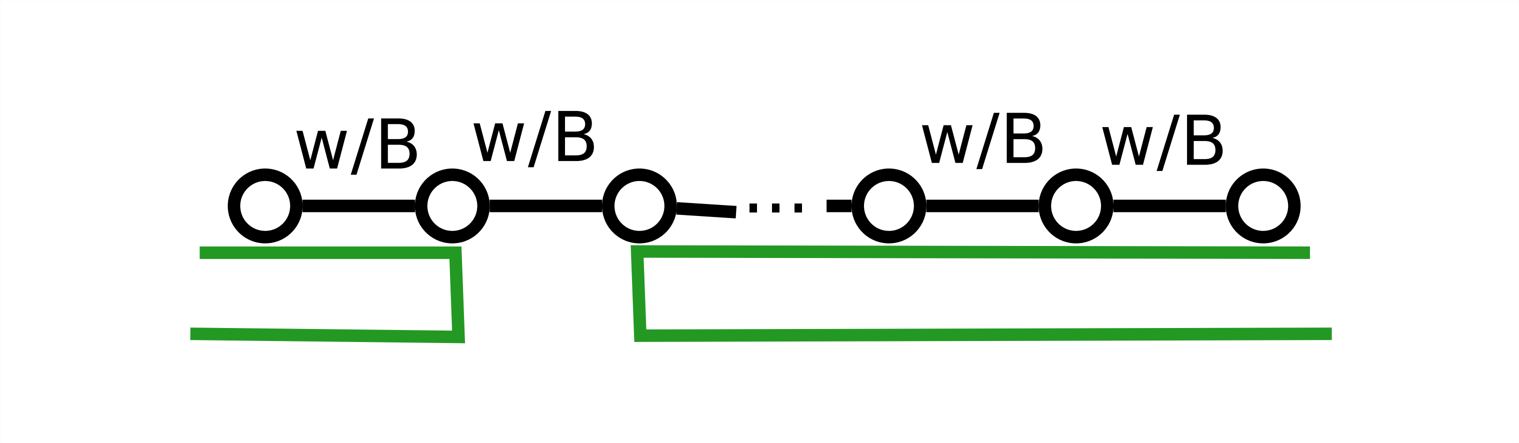
If we had directed edges, this could be achieved by adding a dummy intermediate vertex

Gadget – Forced Edges



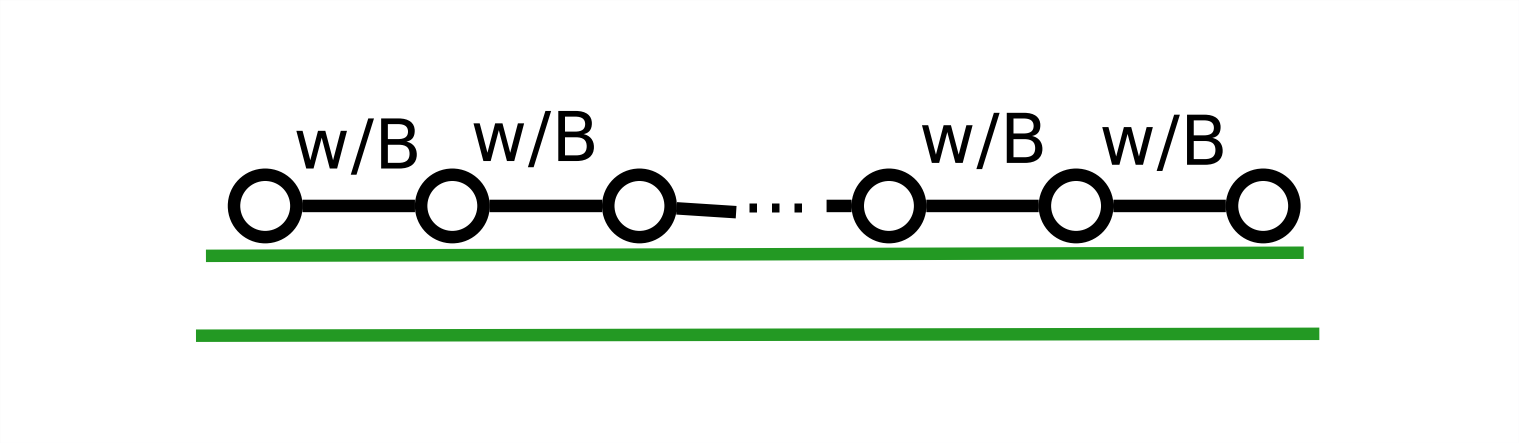
Here, we add many intermediate vertices and evenly distribute the weight w among them. Think of B as very large.

Gadget – Forced Edges



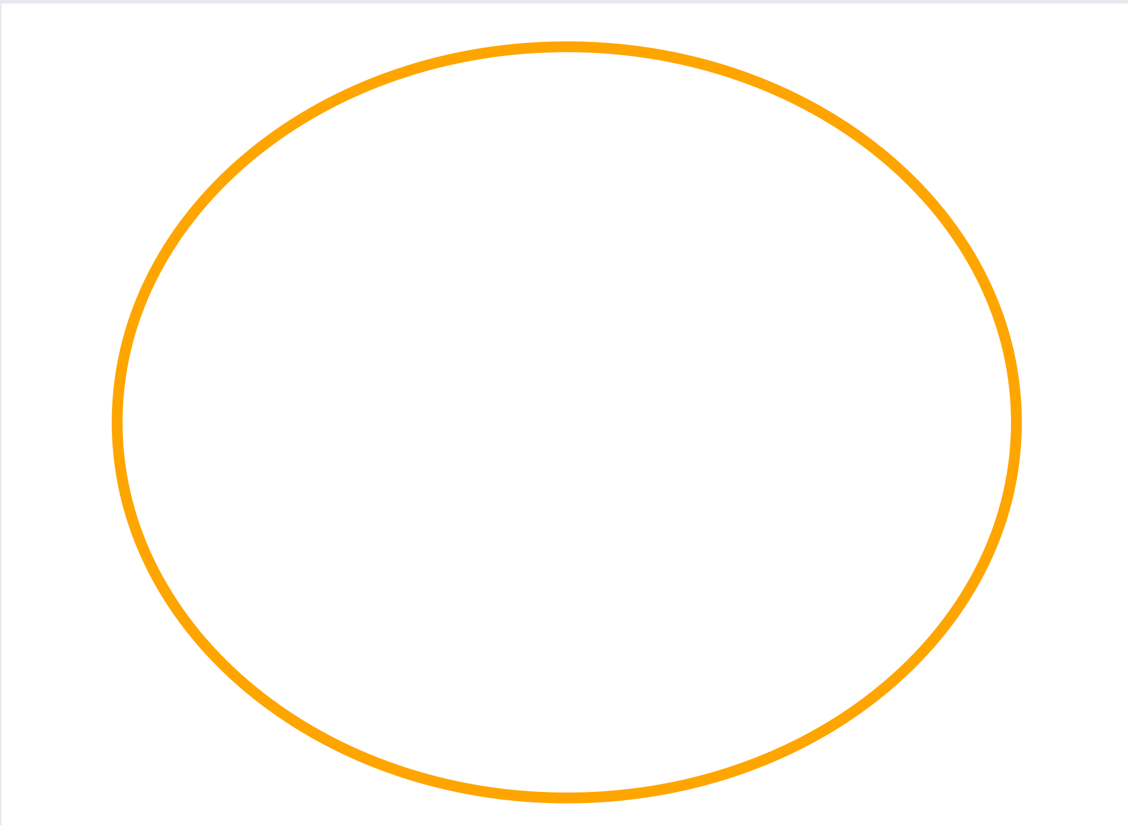
At most one of the new edges may be unused, and in that case all others are used twice.

Gadget – Forced Edges



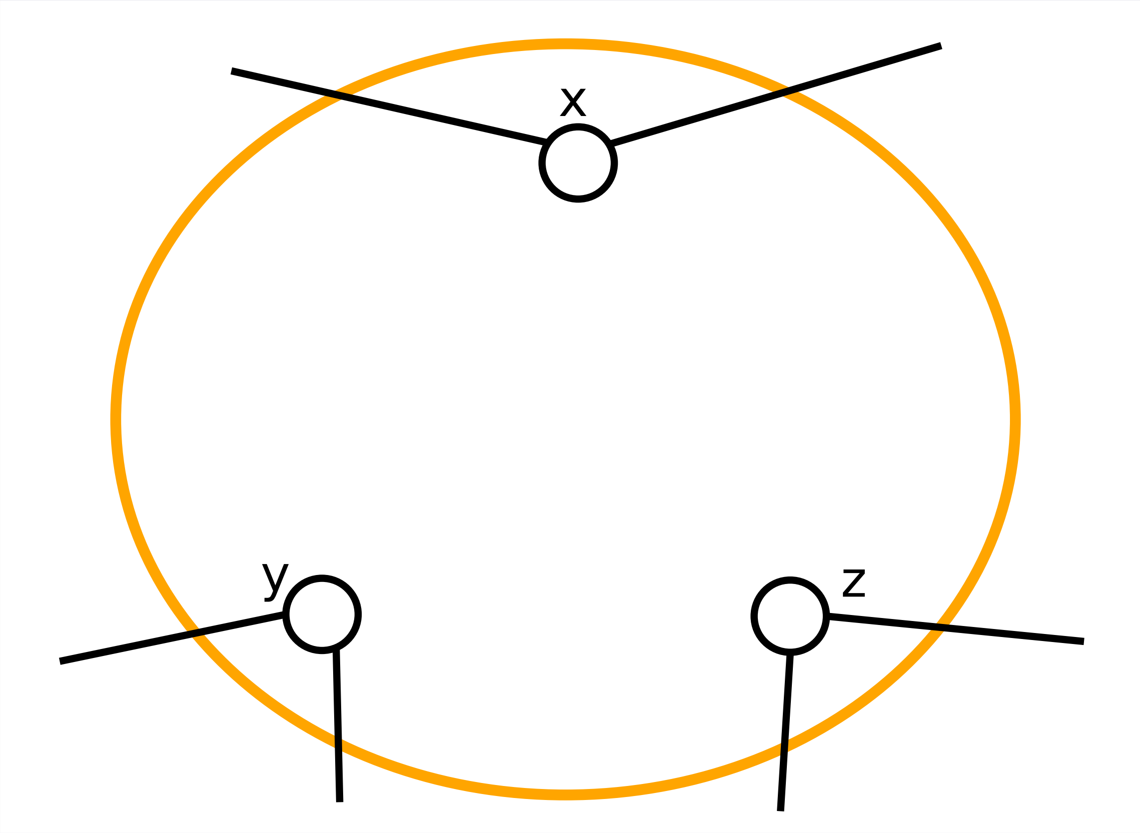
In that case, adding two copies of that edge to the solution doesn't hurt much (for B sufficiently large).

1in3 Gadget



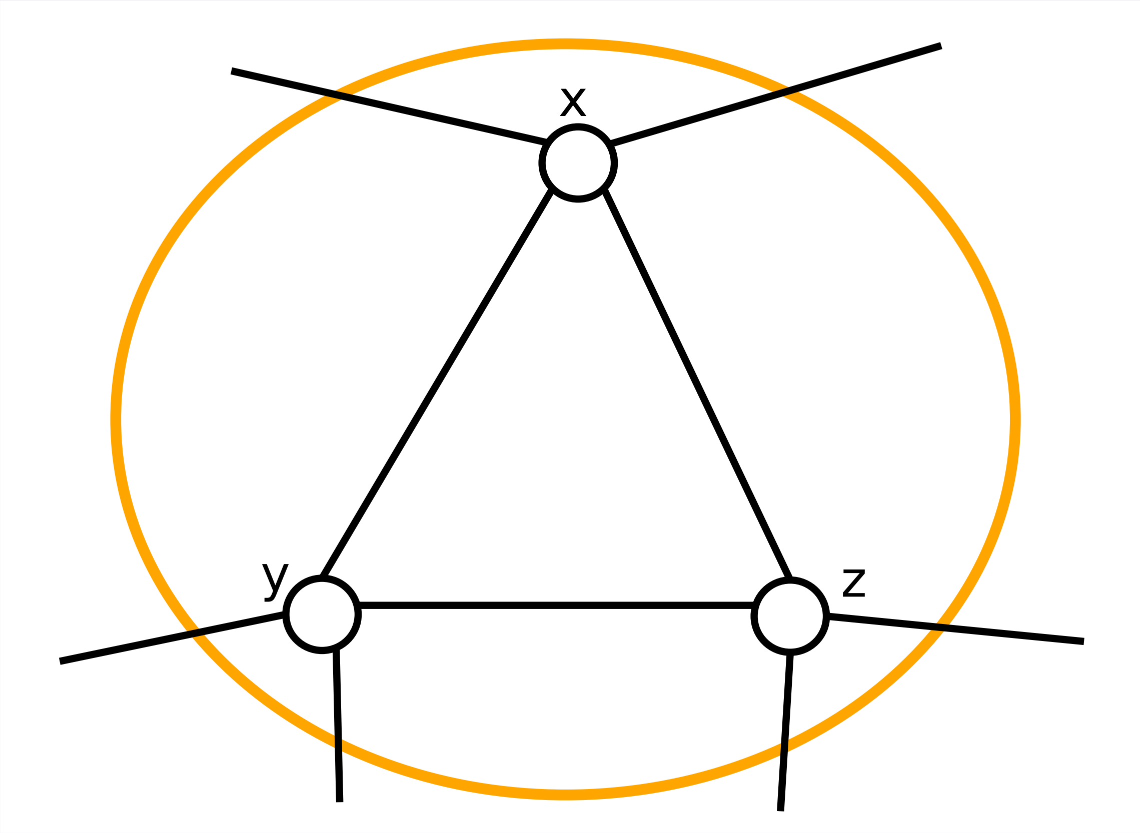
Let's design a gadget for $(x \vee y \vee z)$

1in3 Gadget



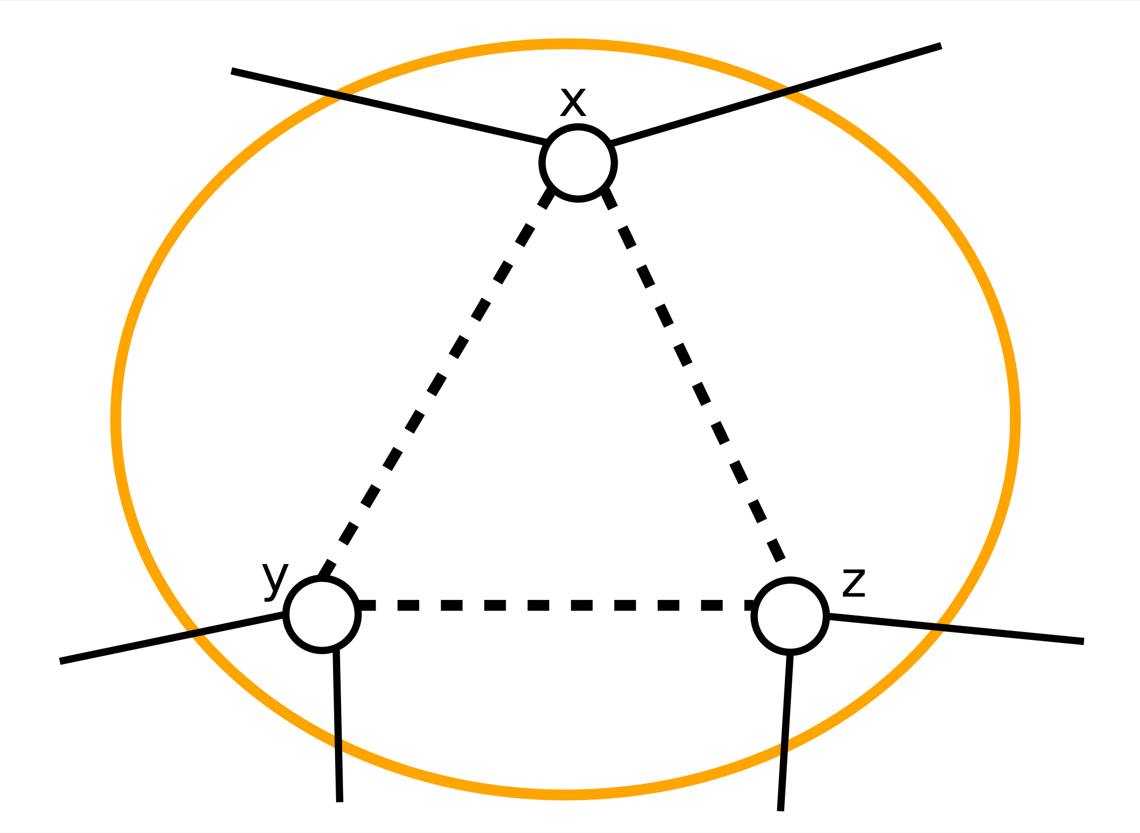
First, three entry/exit points

1in3 Gadget



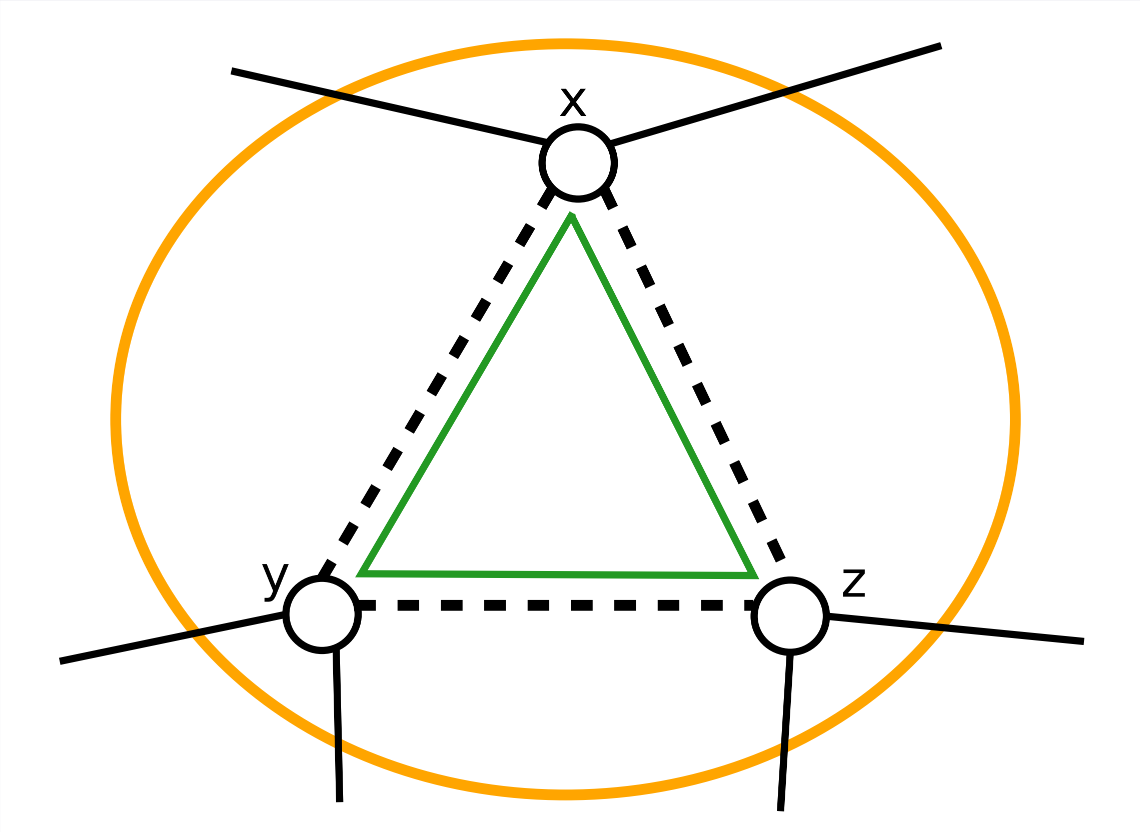
Connect them . . .

1in3 Gadget



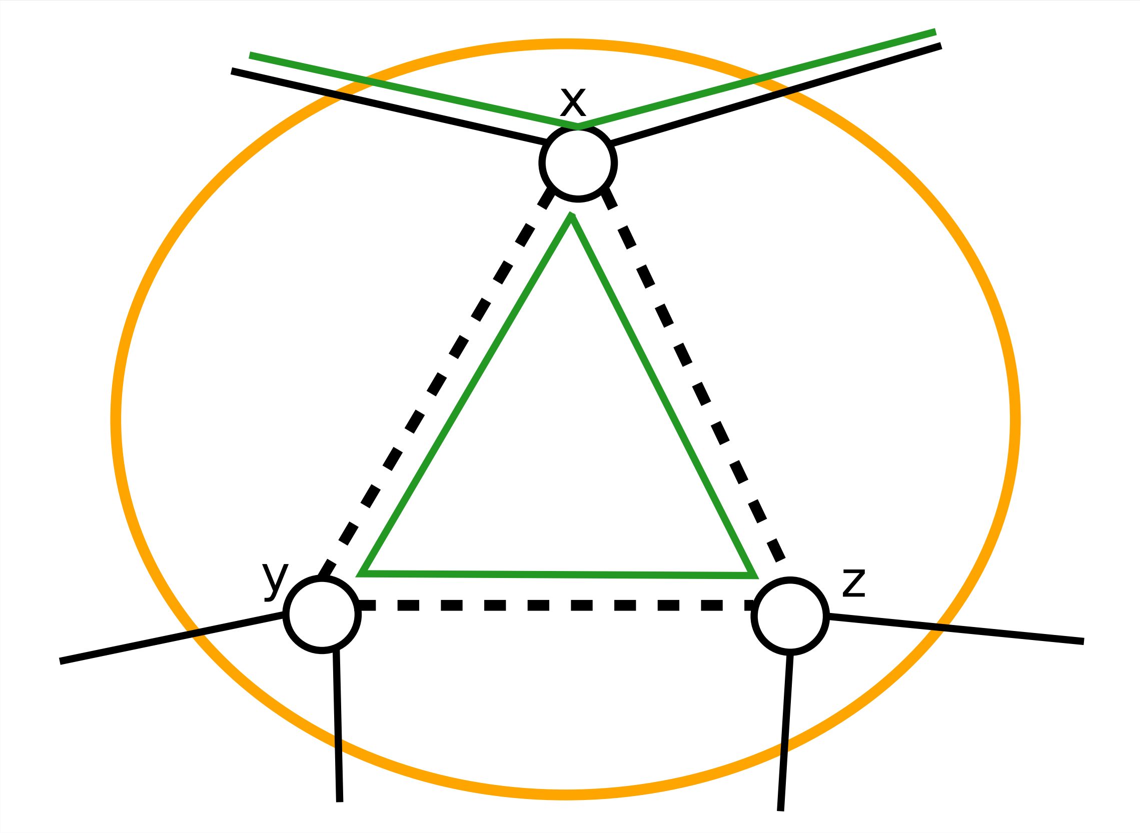
. . . with forced edges

1in3 Gadget



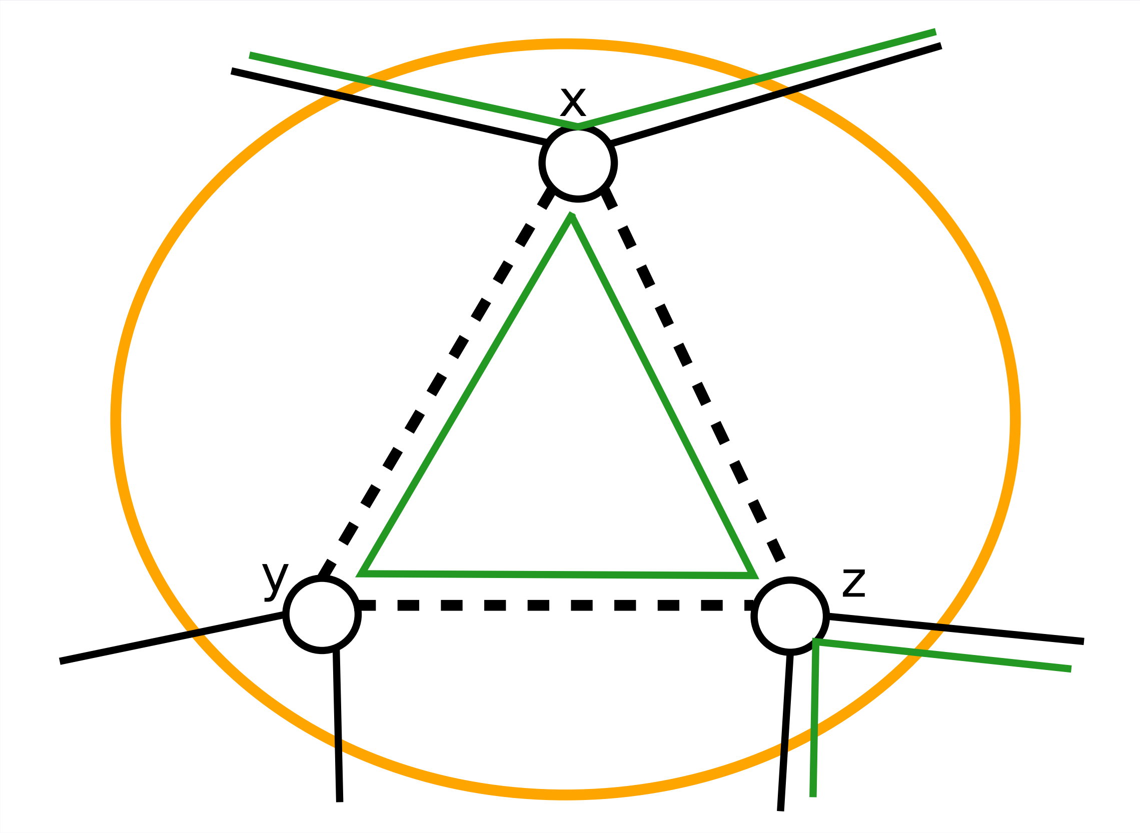
The gadget is a connected component. A good tour visits it once.

1in3 Gadget



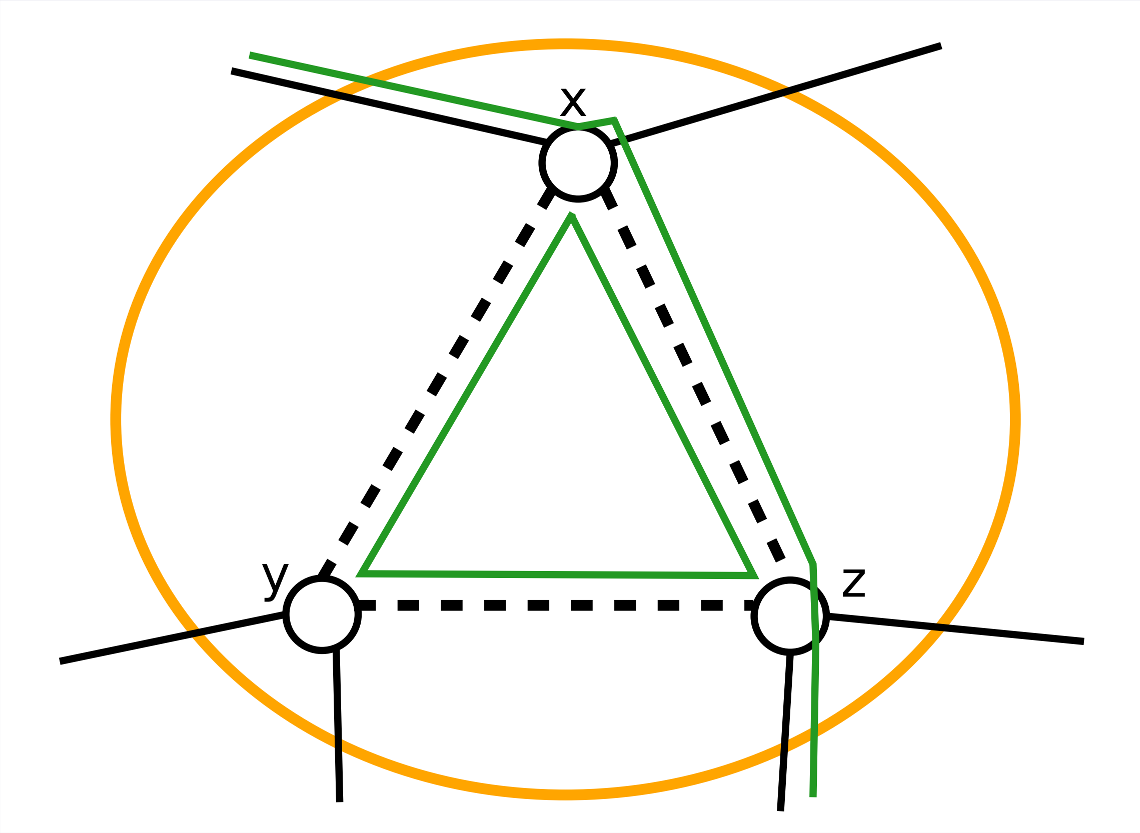
. . . like this

1in3 Gadget

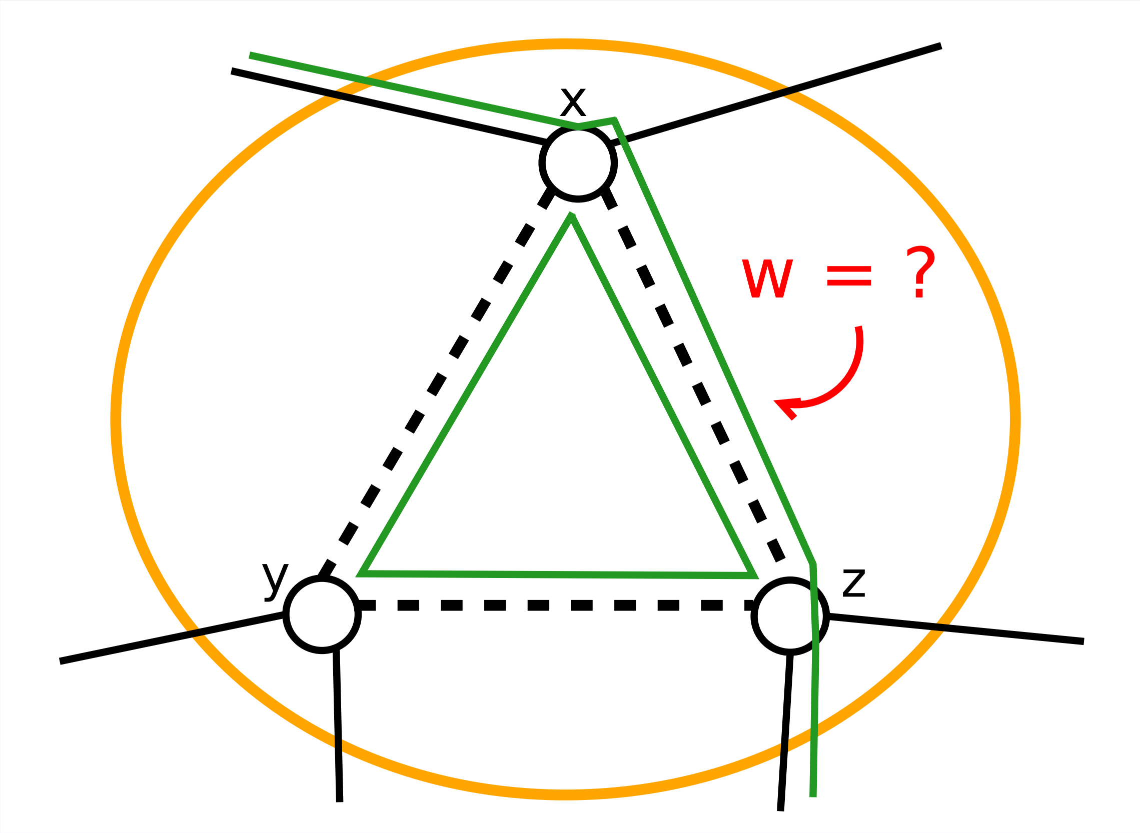


This corresponds to an unsatisfied clause

1in3 Gadget



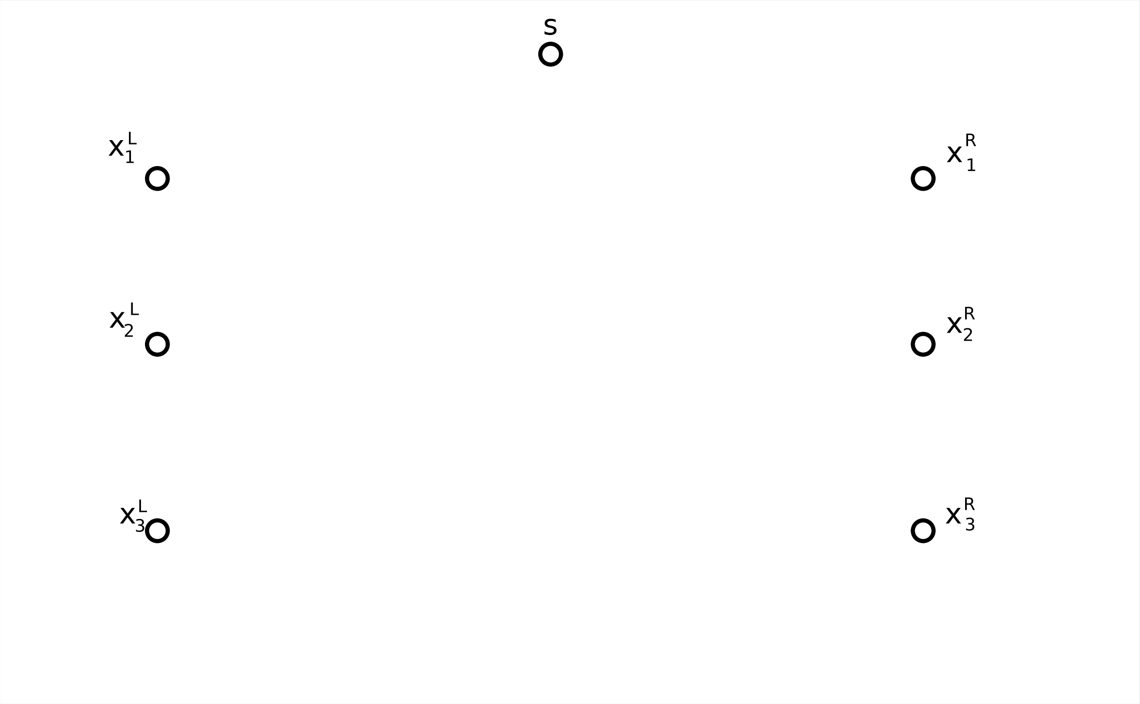
This corresponds to a dishonest tour



The dishonest tour pays this edge twice. How expensive must it be before cheating becomes suboptimal?

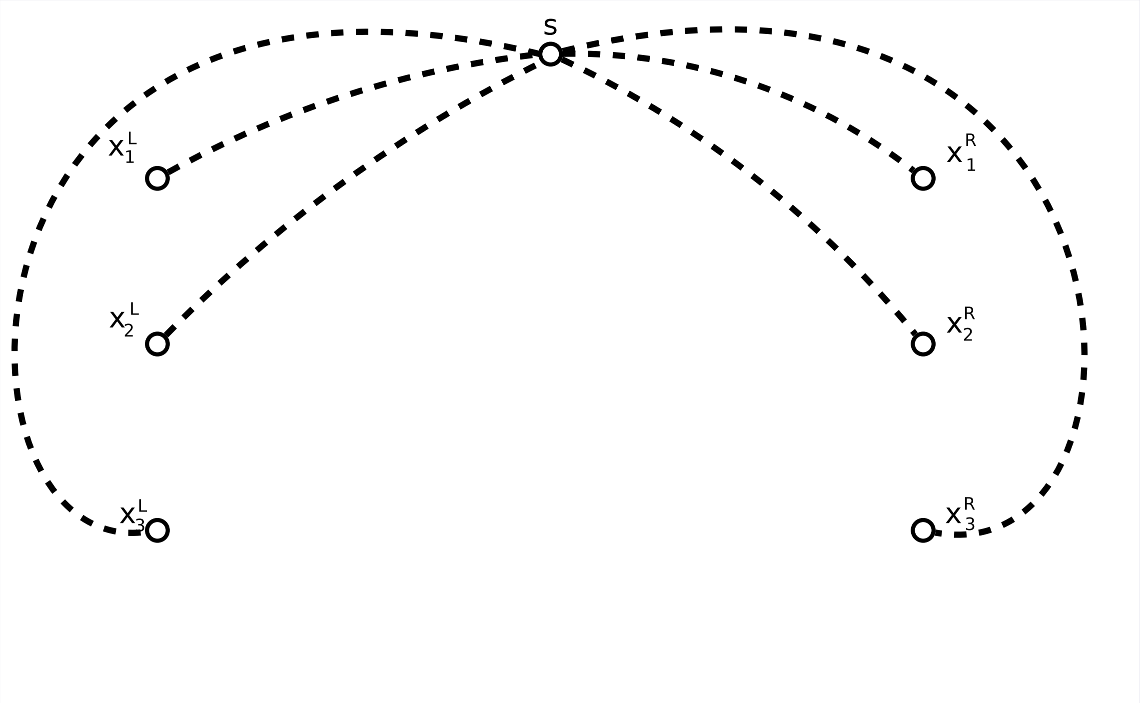
Note that $w = 10$ suffices, since the two cheating variables appear in at most 10 clauses.

Construction



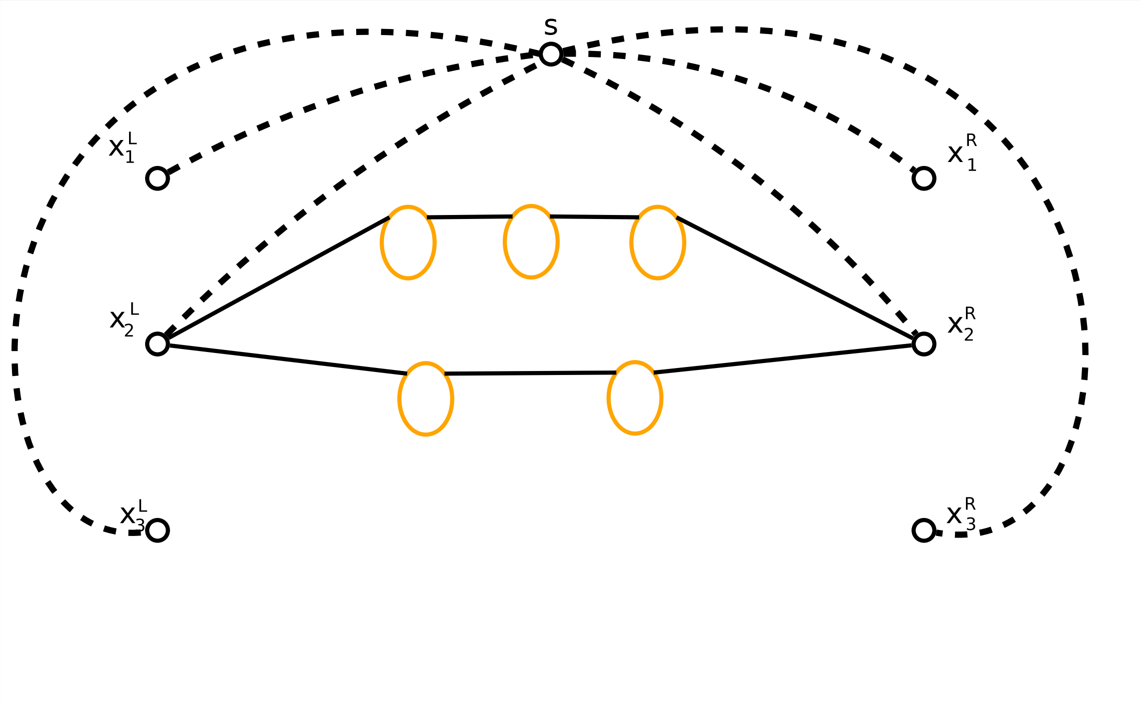
High-level view: construct an origin s and two terminal vertices for each variable.

Construction



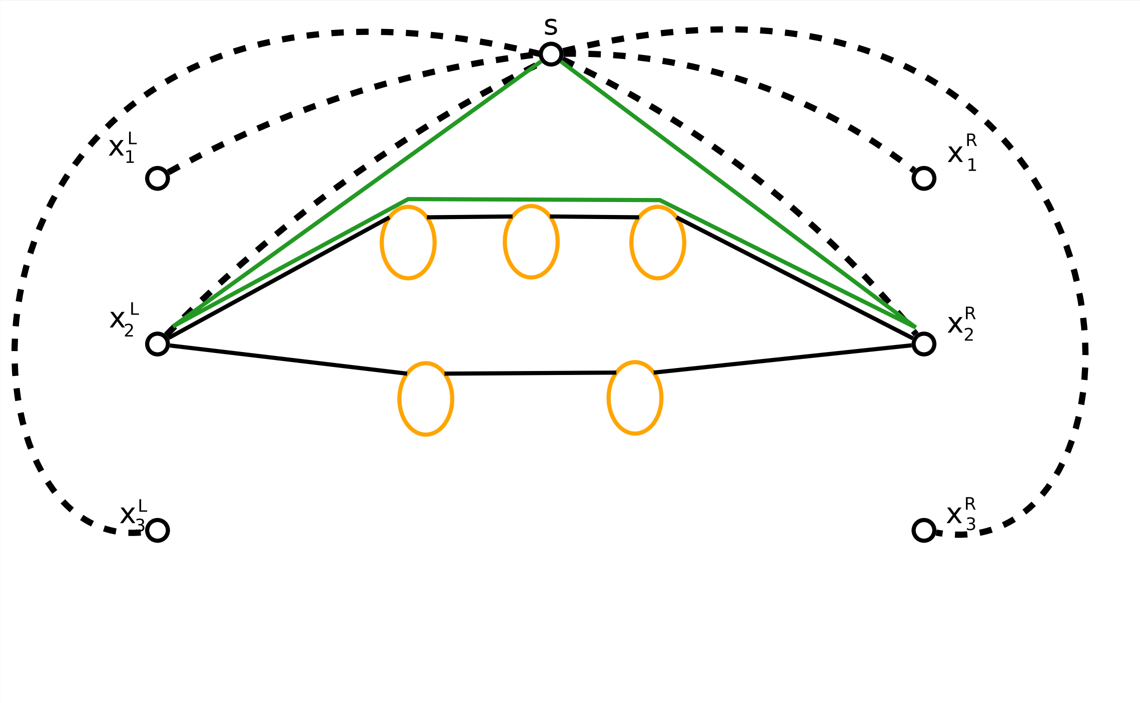
Connect them with forced edges

Construction



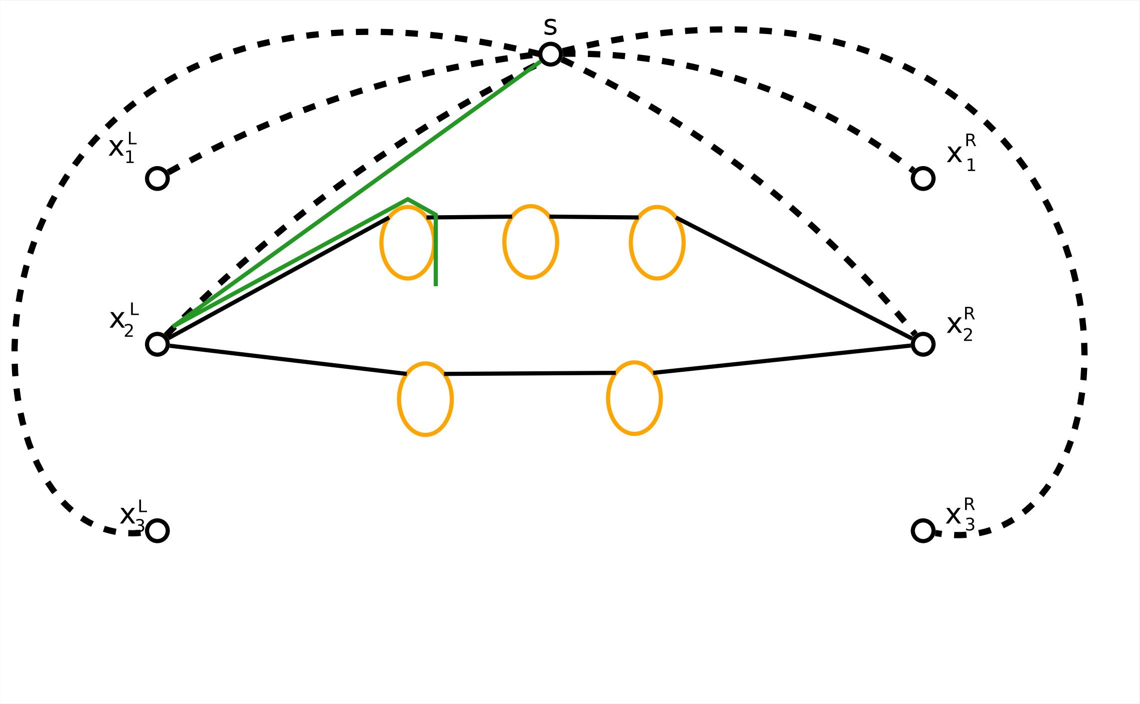
Add the gadgets

Construction

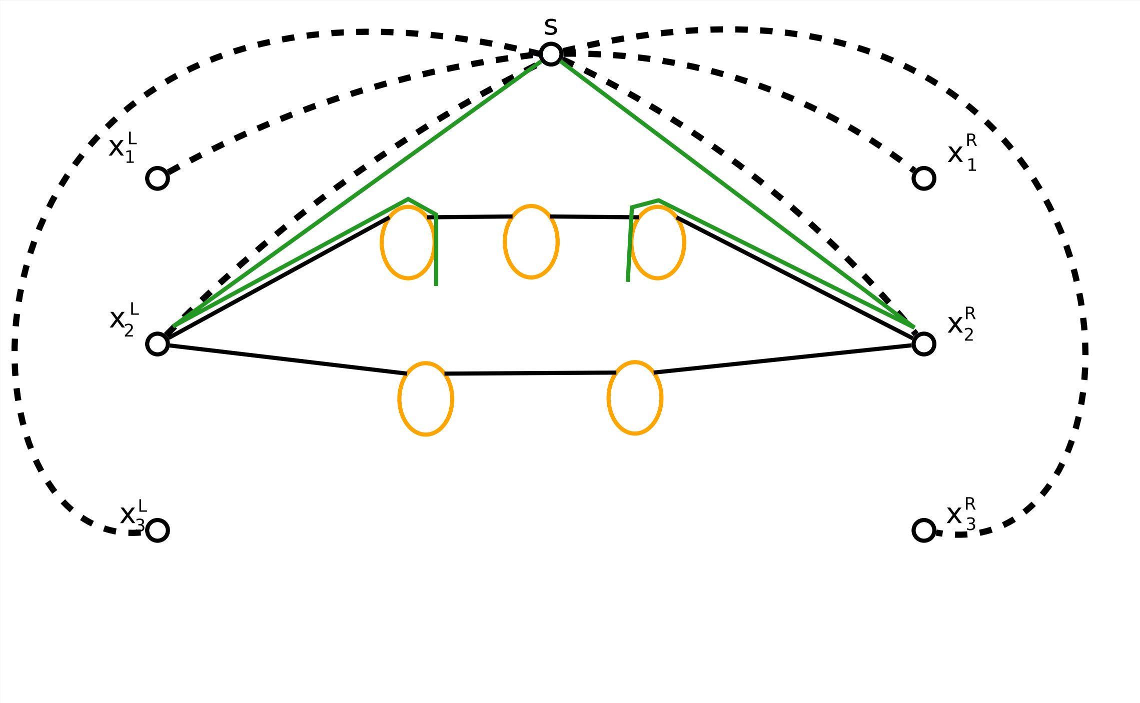


An honest traversal for x_2 looks like this

Construction



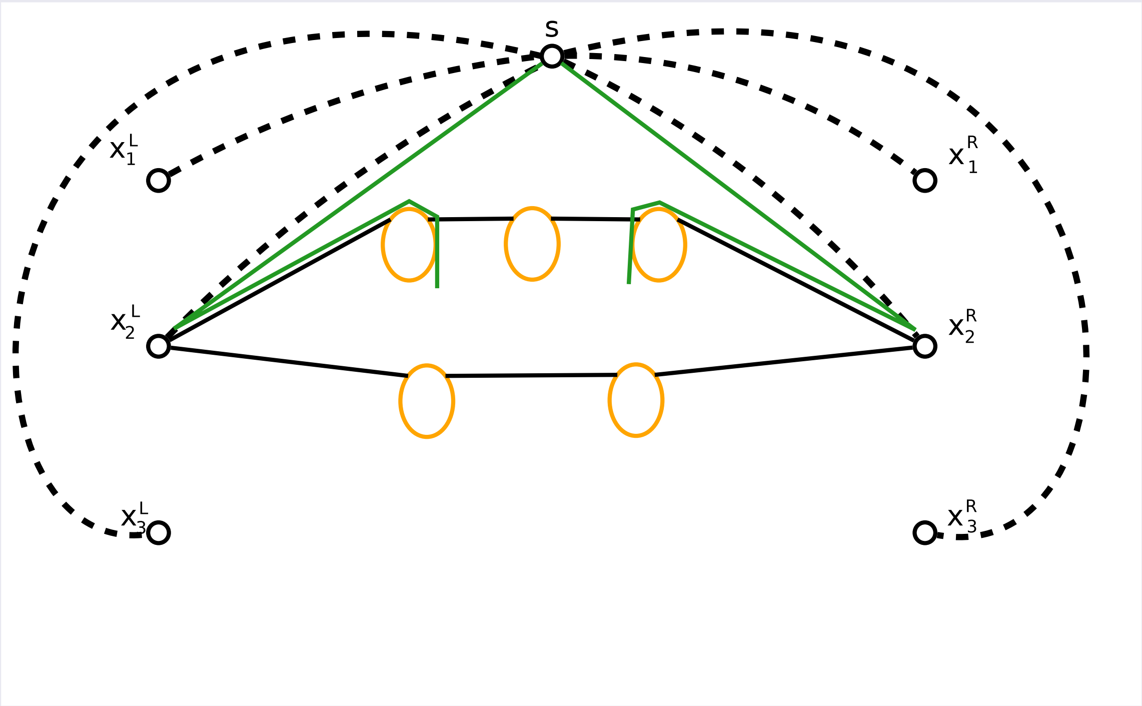
A dishonest traversal looks like this. . .



…but there must be cheating in two places

There are as many doubly-used forced edges as affected variables
$\rightarrow w \leq 5$

Construction



. . . but there must be cheating in two places

There are as many doubly-used forced edges as affected variables
$\rightarrow w \leq 5$

In fact, no need to write off affected clauses. Use random assignment for cheated variables and some of them will be satisfied

Under the carpet

- Many details missing
  - Dishonest variables are set randomly but not independently to ensure that some clauses are satisfied with probability 1.
  - The structure of the instance (from BK amplifier) must be taken into account to calculate the final constant.

Under the carpet

- Many details missing
  - Dishonest variables are set randomly but not independently to ensure that some clauses are satisfied with probability 1.
  - The structure of the instance (from BK amplifier) must be taken into account to calculate the final constant.

Theorem:
There is no $\frac{185}{184}$ approximation algorithm for TSP, unless P=NP.

Under the carpet

- Many details missing
  - Dishonest variables are set randomly but not independently to ensure that some clauses are satisfied with probability 1.
  - The structure of the instance (from BK amplifier) must be taken into account to calculate the final constant.

Theorem:
There is no $\frac{185}{184}$ approximation algorithm for TSP, unless P=NP.

Can we do better?

- Many details missing
  - Dishonest variables are set randomly but not
    clau
  - The
    plifie
    late

Theorem:
There is no $\frac{185}{184}$ approximation algorithm for TSP, unless P NP.

Can we do better?

Update

Bounds have recently been improved further!

- $\frac{123}{122}$ for TSP
- $\frac{75}{74}$ for ATSP

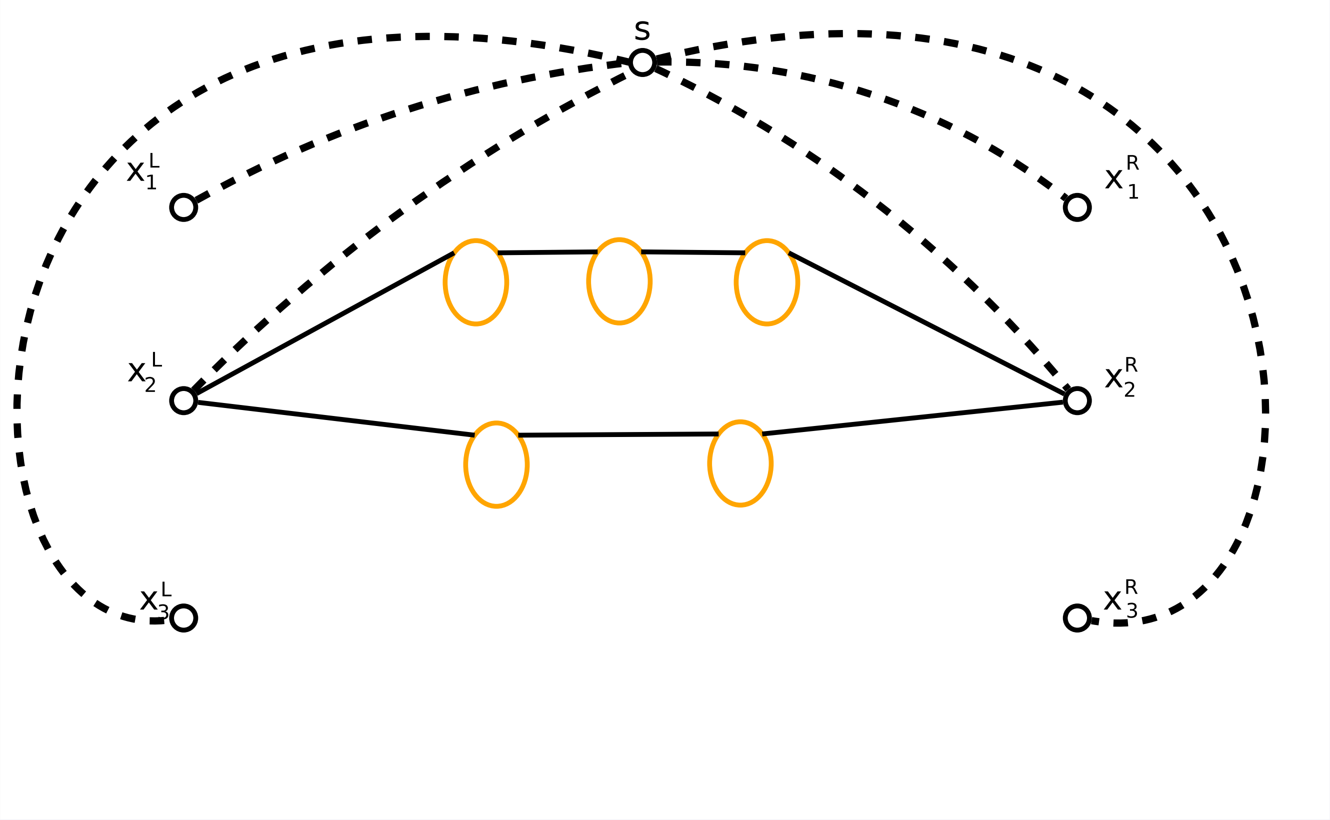
(Joint work with Marek Karpinski, Richard Schmied)

Main ideas:

- Eliminate variable part
- Use 3-regular amplifier
- More clever gadgeteering. . .

Lose the variable part

Recall our construction:



The variable part is pure overhead!

Lose the variable part

We use an idea that:

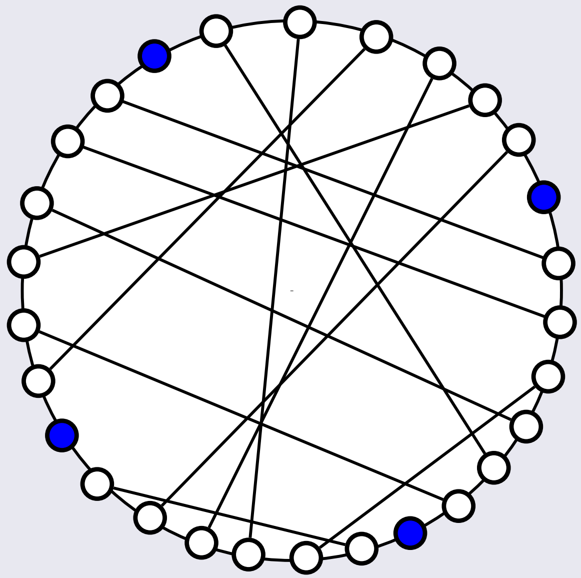
- Eliminates this overhead
- Simulates many of the equations of the amplifier “for free”

Lose the variable part

We use an idea that:

- Eliminates this overhead
- Simulates many of the equations of the amplifier “for free”

- This time we will use the wheel amplifier.
- The idea is to use gadgets only for the matching edges.
- The consistency properties of the gadgets will simulate the cycle edges without extra cost.



Lose the variable part

Construction summary, CSP$\to$TSP:

- For each variable make a vertex
- For each cycle edge make an edge
- Add two gadgets
  - For matching edges
  - For size-three equations

- We are skipping 1-in-3-SAT
- The wheel and the cycle edges are translated unchanged
- Matching edges = inequality gadget from previous reduction

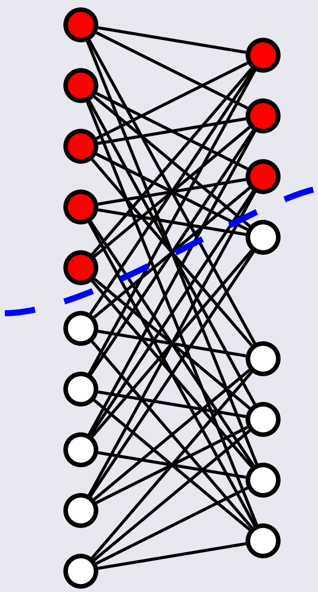
The problem with inequality

- We want to use an inequality gadget to represent the matching edges of the amplifier.
- Normally, amplifier edges become equalities.

The problem with inequality

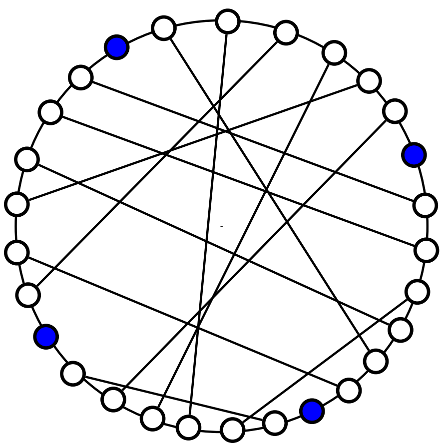
- We want to use an inequality gadget to represent the matching edges of the amplifier.
- Normally, amplifier edges become equalities.

  - Replacing them with inequalities is fine for a bipartite amplifier.



The problem with inequality

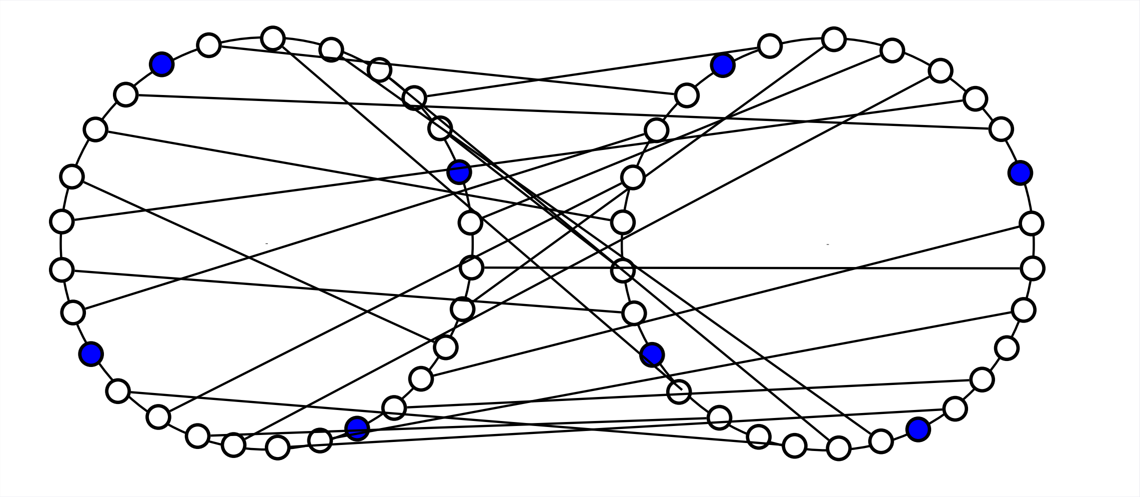
- We want to use an inequality gadget to represent the matching edges of the amplifier.
- Normally, amplifier edges become equalities.



We want cycle edges to remain equalities.

The problem with inequality

- We want to use an inequality gadget to represent the matching edges of the amplifier.
- Normally, amplifier edges become equalities.



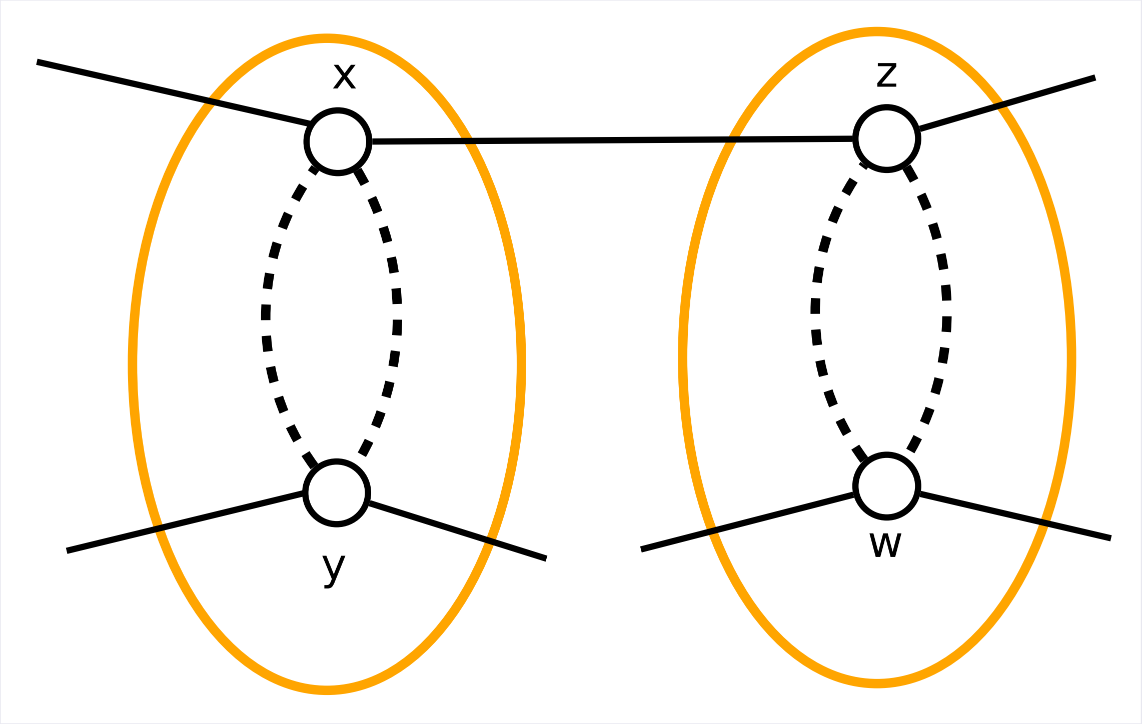
Solution: the bi-wheel!

Free equations!

Main idea: honesty gives equality

Free equations!

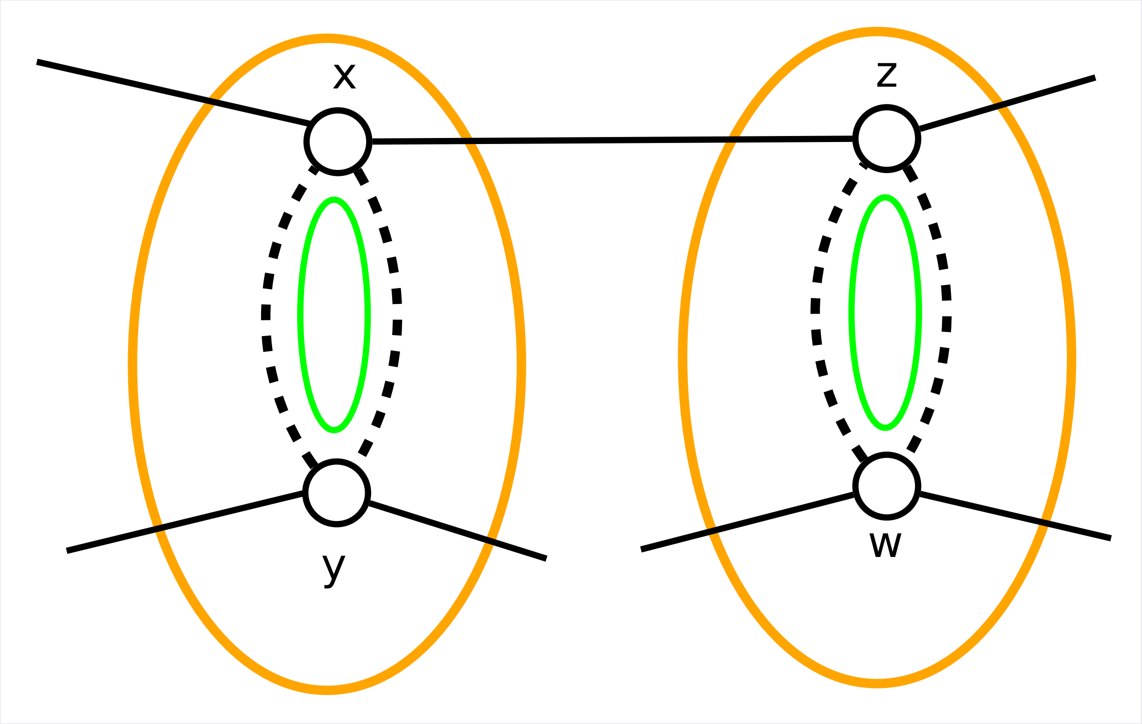
Main idea: honesty gives equality



Consider two vertices consecutive in one cycle (x, z)

Free equations!

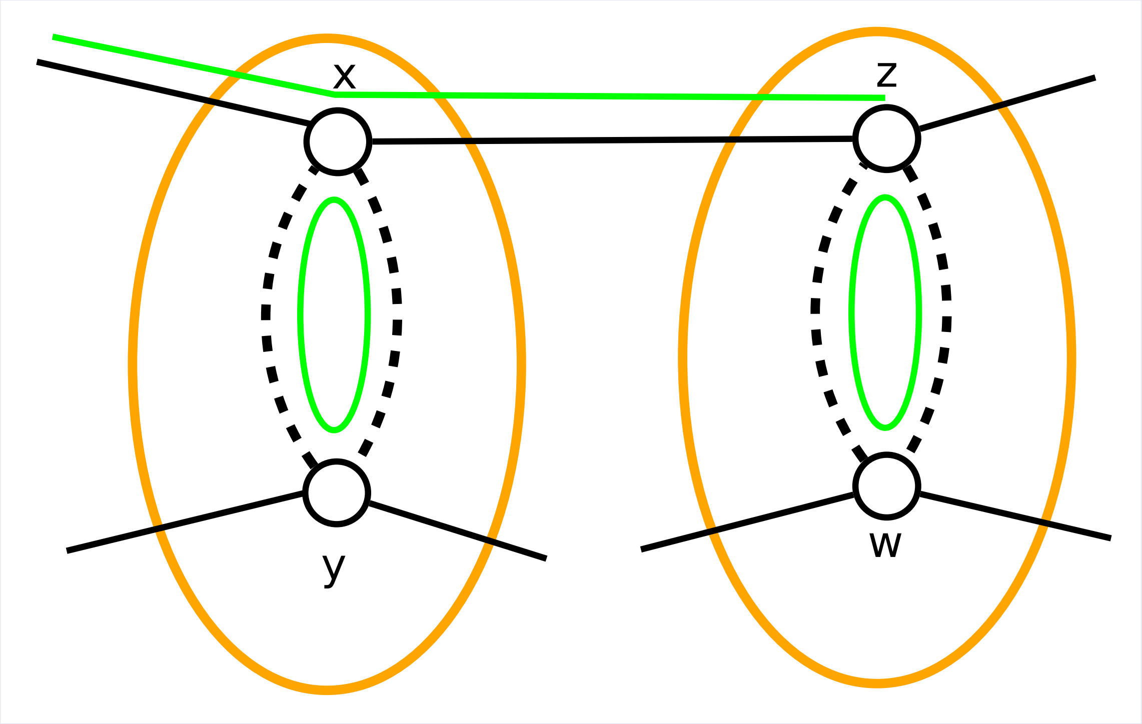
Main idea: honesty gives equality



Suppose that their matching gadgets are honest

Free equations!

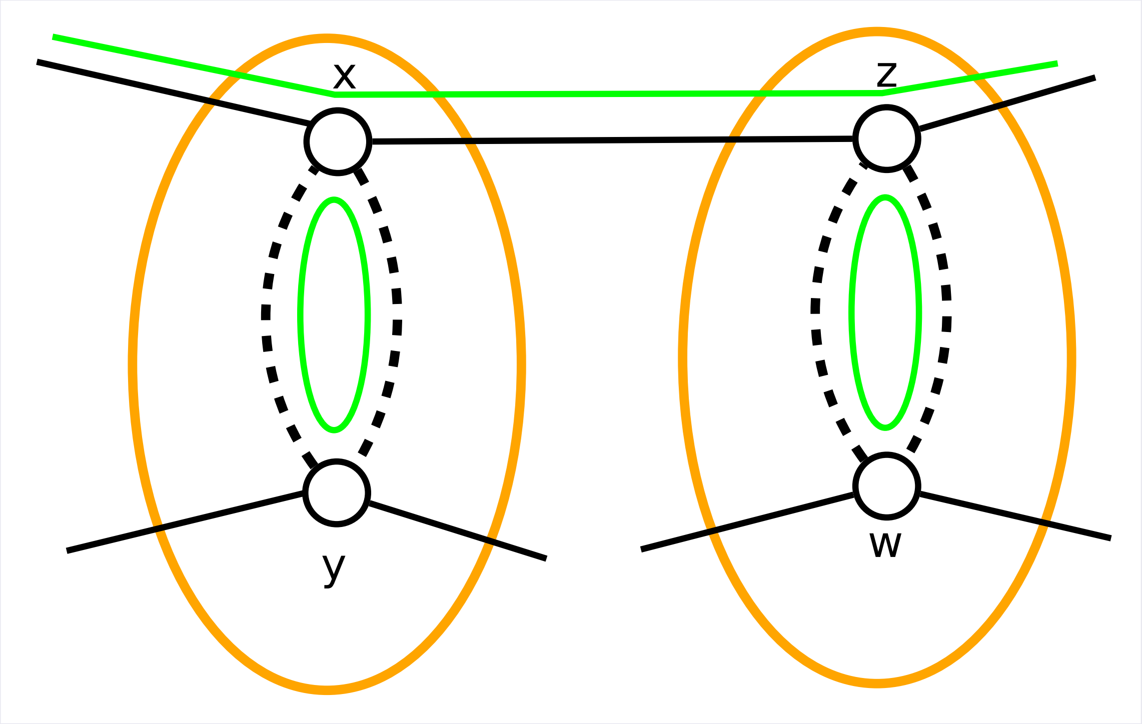
Main idea: honesty gives equality



Then if one is traversed as True. . .

Free equations!

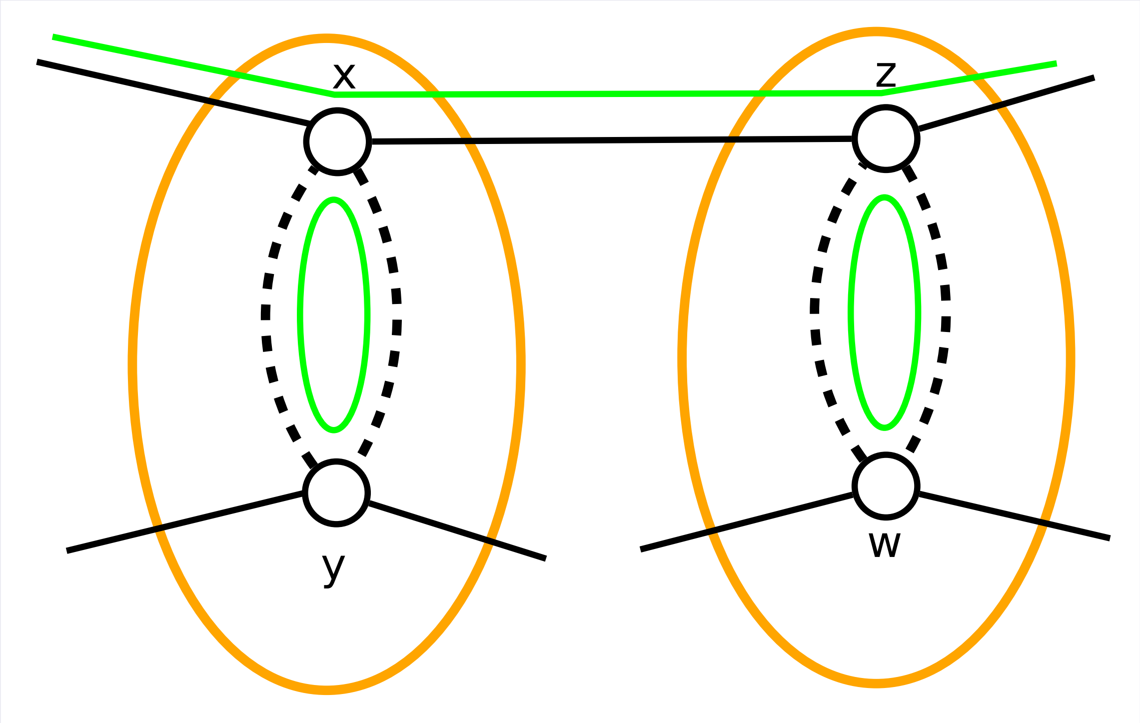
Main idea: honesty gives equality



. . . the other is also!

Free equations!

Main idea: honesty gives equality

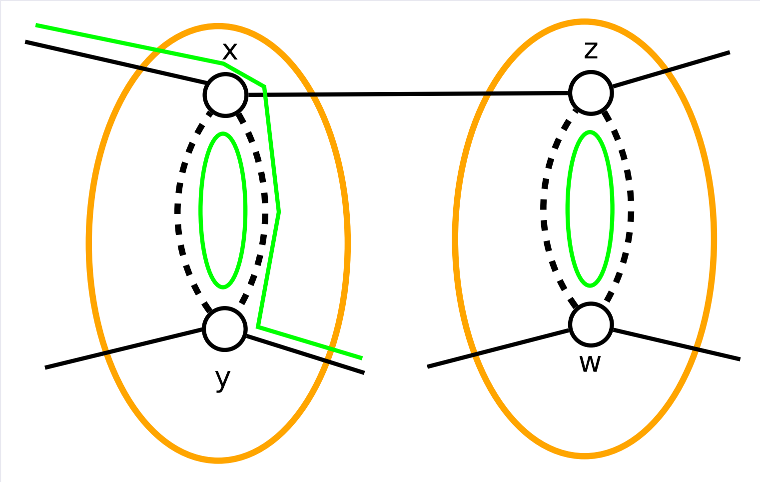


. . . the other is also!

- In other words, we extract an assignment for x by setting it to 1 iff both its incident non-forced edges are used.

Some handwaving

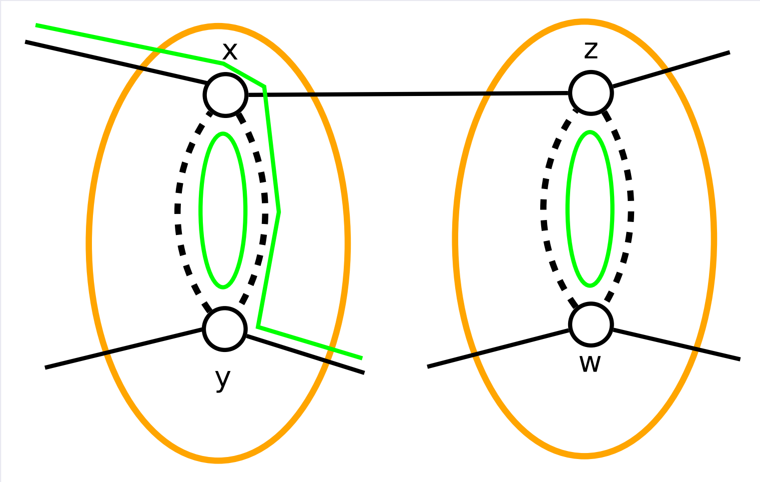
What is the cost of the forced edges?



- In case of dishonest traversal we must make the tour pay for all unsatisfied equations.

Some handwaving

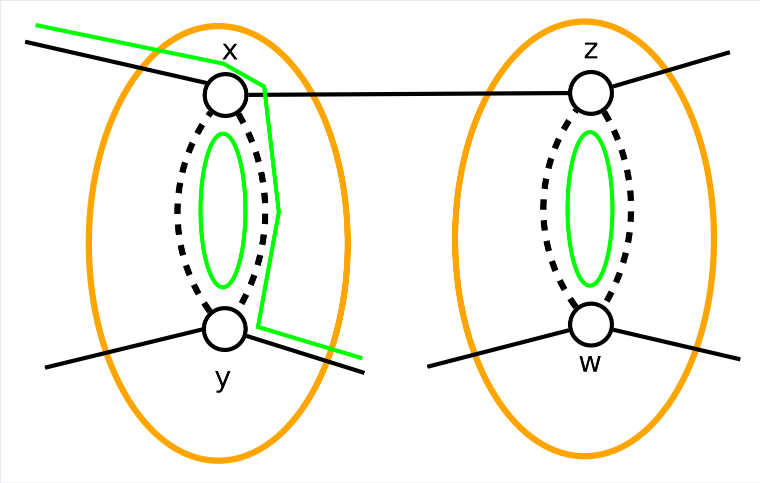
What is the cost of the forced edges?



- In case of dishonest traversal we must make the tour pay for all unsatisfied equations.
- There are 5 affected equation.

Some handwaving

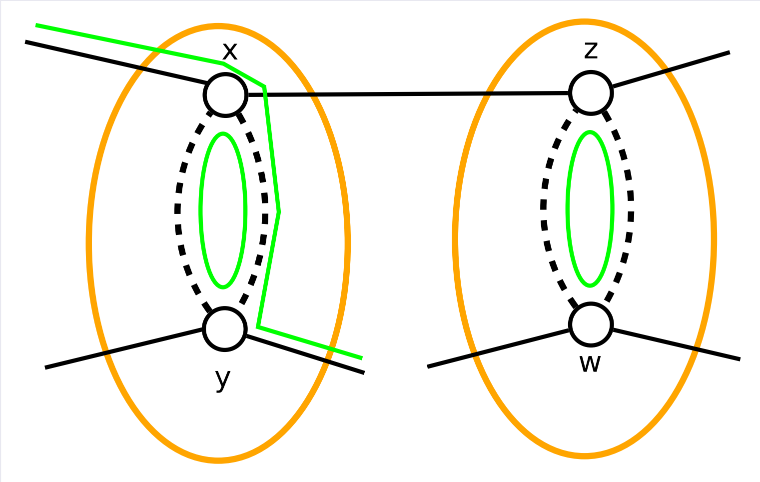
What is the cost of the forced edges?



- In case of dishonest traversal we must make the tour pay for all unsatisfied equations.
- There are 5 affected equation.
- We can always satisfy 3.

Some handwaving

What is the cost of the forced edges?



- In case of dishonest traversal we must make the tour pay for all unsatisfied equations.
- There are 5 affected equation.
- We can always satisfy 3.
- Hence, cost of forced edges is 2.

More handwaving

- For size-three equations we come up with some gadget (not shown).
- Some work needs to be done to ensure connectivity.
- Similar ideas can be used for ATSP.

More handwaving

- For size-three equations we come up with some gadget (not shown).
- Some work needs to be done to ensure connectivity.
- Similar ideas can be used for ATSP.

Theorem:
There is no $\frac{123}{122} - \epsilon$ approximation algorithm for TSP, unless P=NP.
There is no $\frac{75}{74} - \epsilon$ approximation algorithm for ATSP, unless P=NP.

Conclusions – Open problems

- A simpler reduction for TSP and a better inapproximability threshold
  - But, constant still very low!

Future work

- Better amplifier constructions?
- Application for improved expanders?

Conclusions – Open problems

- A simpler reduction for TSP and a better inapproximability threshold
    - But, constant still very low!

Future work

- Better amplifier constructions?
- Application for improved expanders?
- . . . **Reasonable** inapproximability for TSP?

The end

Questions?