Coping with NP-completeness: Approximation Algorithms

Alexander S. Kulikov

Steklov Institute of Mathematics at St. Petersburg
Russian Academy of Sciences

Advanced Algorithms and Complexity
Data Structures and Algorithms

Outline

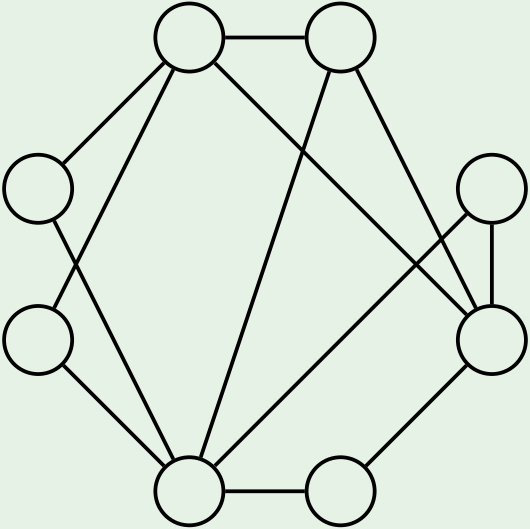
1. Vertex cover
2. Traveling salesman
   - Metric TSP
   - Local search

Vertex cover (optimization version)

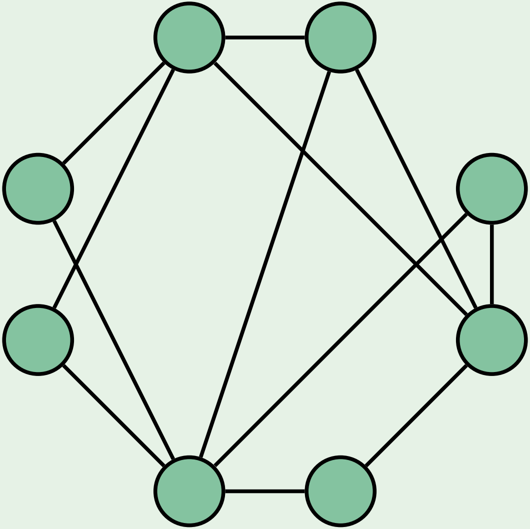
Input: A graph.

Output: A subset of vertices of minimum size that touches every edge.

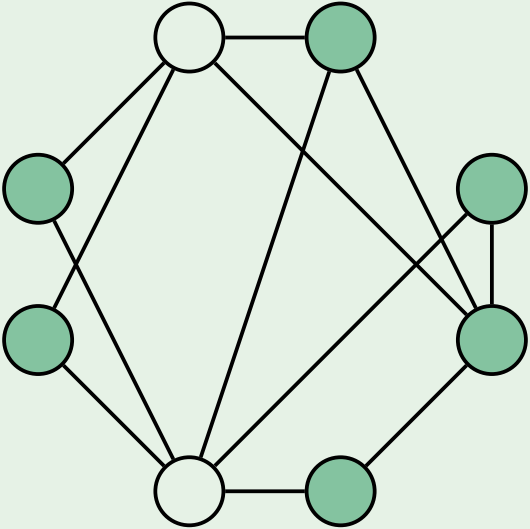
Example



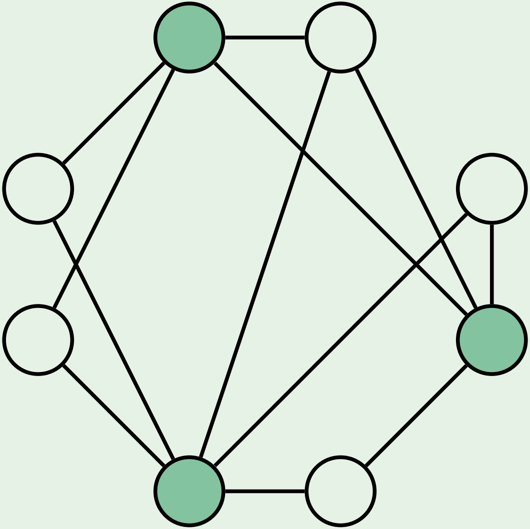
Example



Example



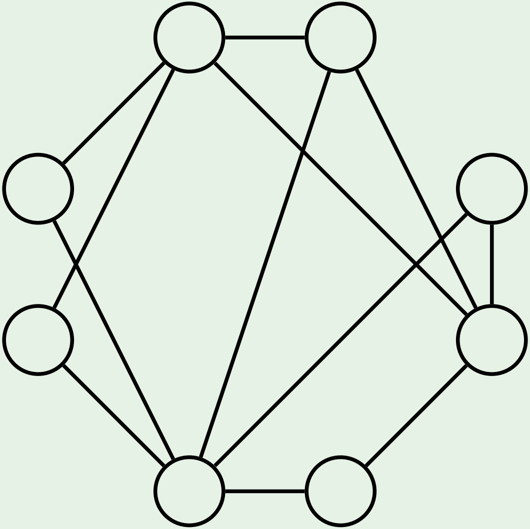
Example



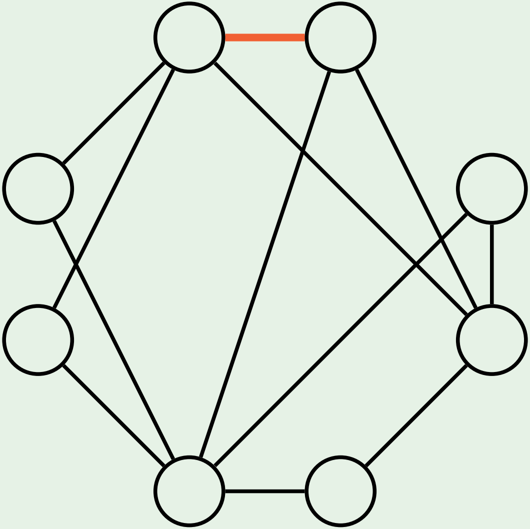
ApproxVertexCover($G(V, E)$)

```
C ← empty set
while E is not empty:
  {u, v} ← any edge from E
  add u, v to C
  remove from E all edges incident to u, v
return C
```

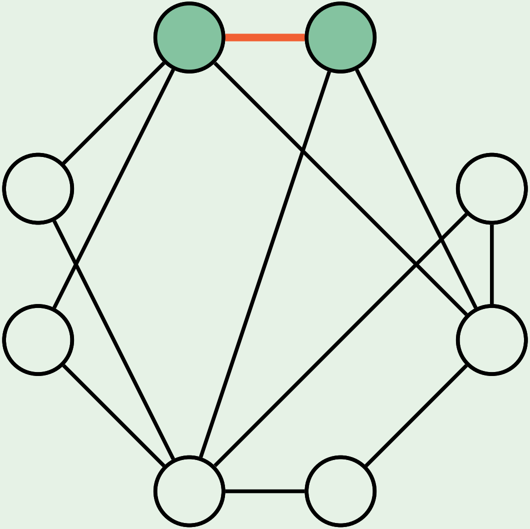
Example



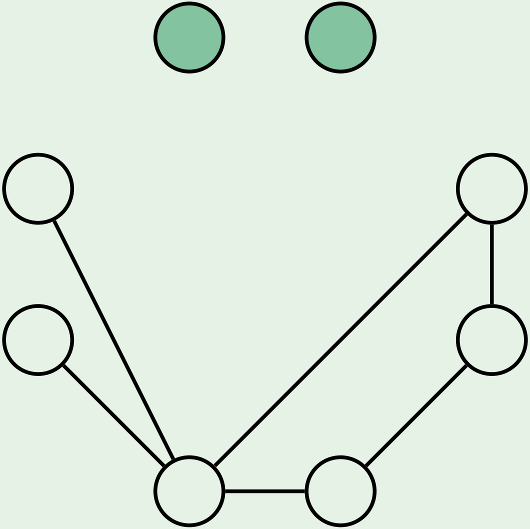
Example



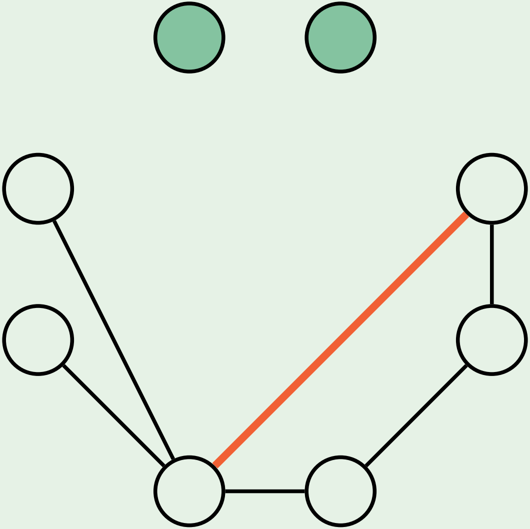
Example



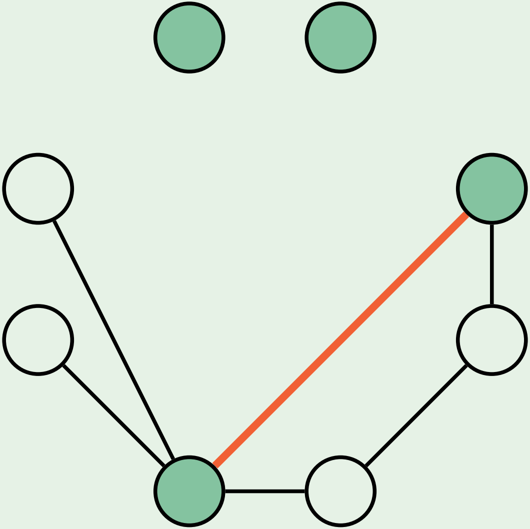
Example



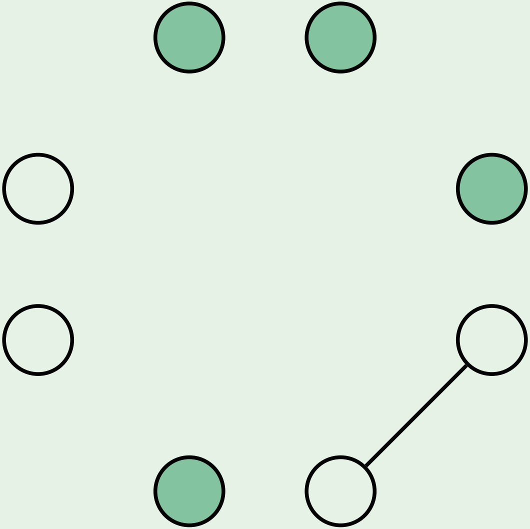
Example



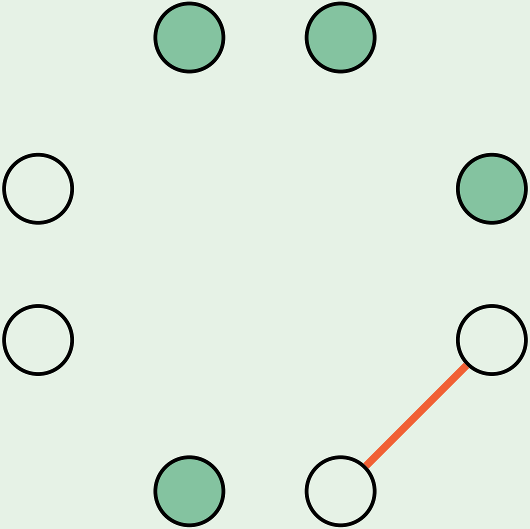
Example



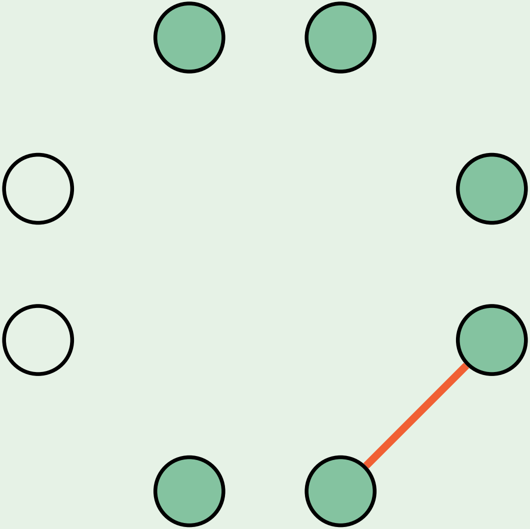
Example



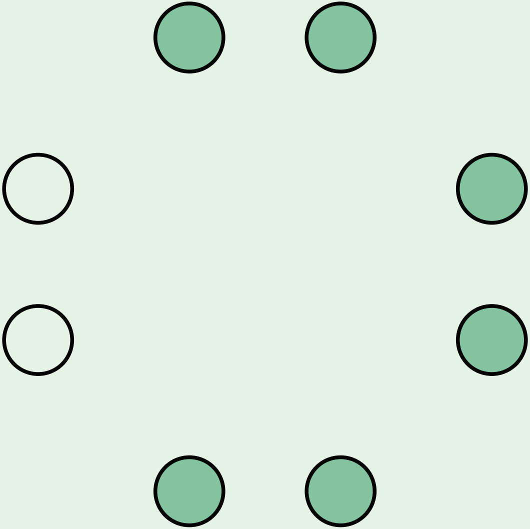
Example



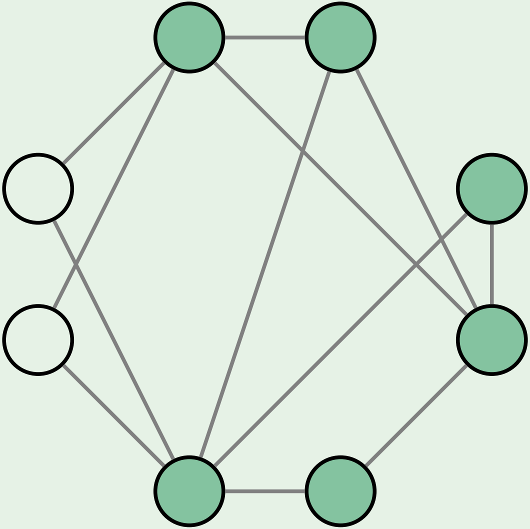
Example



Example



Example



Lemma

The algorithm `ApproxVertexCover` is 2-approximate: it returns a vertex cover that is at most twice as large as an optimal one and runs in polynomial time.

Proof

- The set M of all edges selected by the algorithm forms a matching

Proof

- The set M of all edges selected by the algorithm forms a matching
- Any vertex cover of the graph has size at least $|M|$

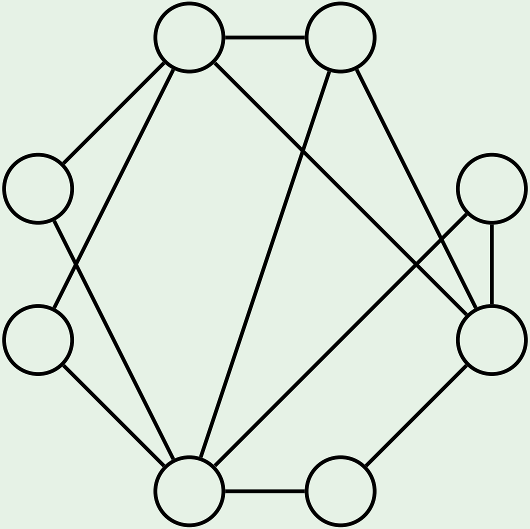
Proof

- The set M of all edges selected by the algorithm forms a matching
- Any vertex cover of the graph has size at least $|M|$
- The algorithm returns a vertex cover C of size $2|M|$, hence

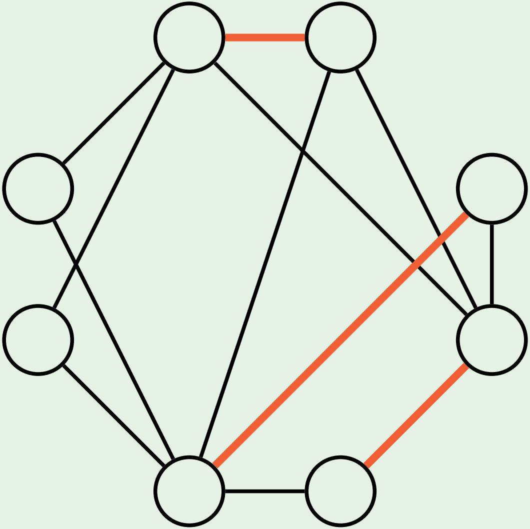
$$|C| = 2 \cdot |M| \leq 2 \cdot \mathrm{OPT}$$

□

Example



Example



Summary

- We don't know the value of OPT, but we've managed to prove that

$$|C| \leq 2 \cdot \mathrm{OPT}$$

Summary

- We don't know the value of OPT, but we've managed to prove that

$$|C| \leq 2 \cdot \mathrm{OPT}$$

- This is because we know a lower bound on OPT: it is at least the size of any matching

$$|C| = 2 \cdot |M| \leq 2 \cdot \mathrm{OPT}$$

Final Remarks

- The bound is tight: there are graphs for which the algorithm returns a vertex cover of size twice the minimum size.

Final Remarks

- The bound is tight: there are graphs for which the algorithm returns a vertex cover of size twice the minimum size.
- No 1.99-approximation algorithm is known.

Outline

1. Vertex cover
2. Traveling salesman
   - Metric TSP
   - Local search

Outline

1. Vertex cover
2. Traveling salesman
    - Metric TSP
    - Local search

Metric TSP (optimization version)

Input: An undirected graph $G(V, E)$ with non-negative edge weights satisfying the triangle inequality: for all $u, v, w \in V$, $d(u, v) + d(v, w) \geq d(u, w)$.

Output: A cycle of minimum total length visiting each vertex exactly once .

Lower Bound

- We are going to design a 2-approximation algorithm: it returns a cycle that is at most twice as long as an optimal cycle: $C \leq 2 \cdot \mathrm{OPT}$

Lower Bound

- We are going to design a 2-approximation algorithm: it returns a cycle that is at most twice as long as an optimal cycle: $C \leq 2 \cdot \mathrm{OPT}$
- Since we don't know the value of OPT, we need a good lower bound L on OPT:

$$C \leq 2 \cdot L \leq 2 \cdot \mathrm{OPT}$$

Minimum Spanning Trees

Lemma

Let G be an undirected graph with non-negative edge weights. Then $\mathsf{MST}(G) \leq \mathsf{TSP}(G)$.

Minimum Spanning Trees

Lemma

Let G be an undirected graph with non-negative edge weights. Then $\mathrm{MST}(G) \leq \mathrm{TSP}(G)$.

Proof

By removing any edge from an optimum TSP cycle one gets a spanning tree of G. $\square$

ApproxMetricTSP(G)

$T \leftarrow$ `minimum spanning tree of` G

ApproxMetricTSP(G)

$T \leftarrow$ `minimum spanning tree of` G
$D \leftarrow T$ `with each edge doubled`

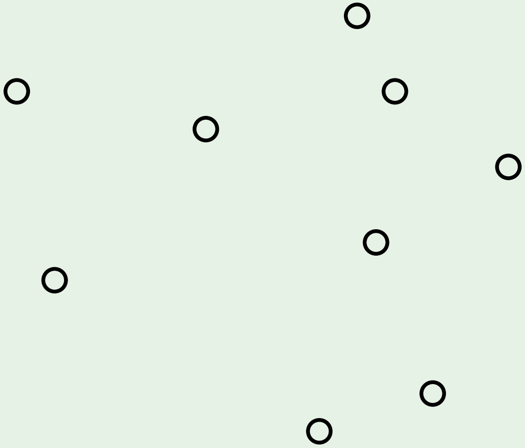
ApproxMetricTSP(G)

$T \leftarrow$ minimum spanning tree of G
$D \leftarrow T$ with each edge doubled
find an Eulerian cycle C in D

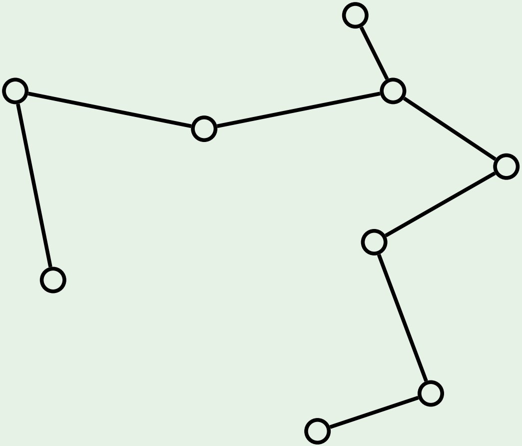
ApproxMetricTSP(G)

$T \leftarrow$ minimum spanning tree of G
$D \leftarrow T$ with each edge doubled
find an Eulerian cycle C in D
return a cycle that visits vertices in
  the order of their first appearance in C

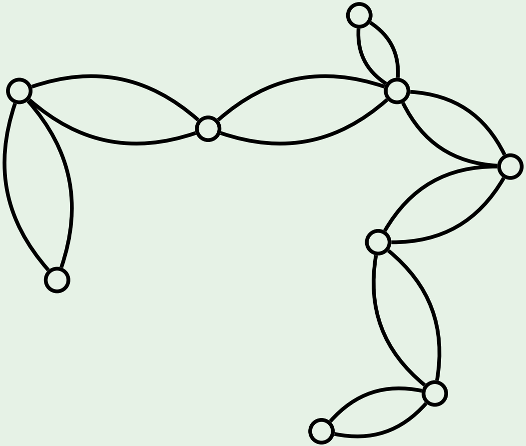
Example: points on a plane



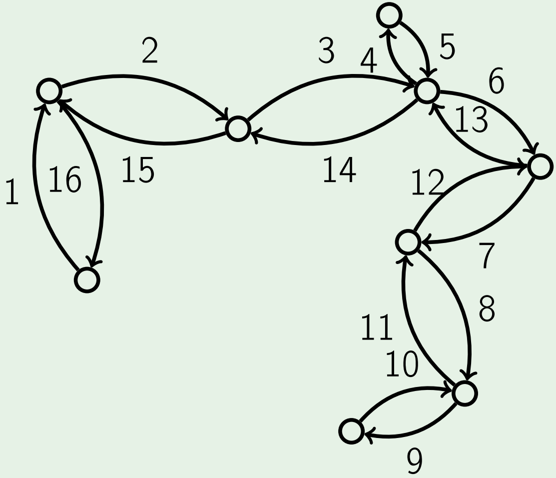
Example: points on a plane



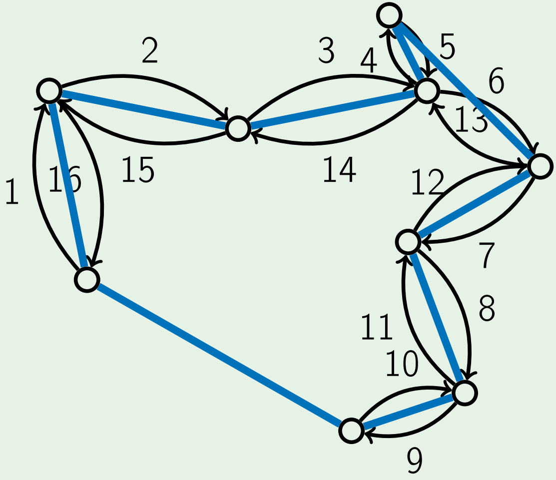
Example: points on a plane



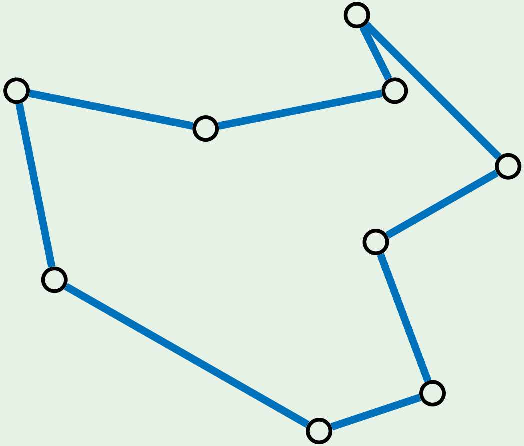
Example: points on a plane



Example: points on a plane



Example: points on a plane



Lemma

The algorithm `ApproxMetricTSP` is 2-approximate.

Lemma

The algorithm `ApproxMetricTSP` is 2-approximate.

Proof

- The total length of the MST T is at most OPT.

Lemma

The algorithm `ApproxMetricTSP` is 2-approximate.

Proof

- The total length of the MST T is at most OPT.
- Bypasses can only decrease the total length. $\square$

Final Remarks

- The currently best known approximation algorithm for metric TSP is Christofides' algorithm that achieves a factor of 1.5

Final Remarks

- The currently best known approximation algorithm for metric TSP is Christofides' algorithm that achieves a factor of 1.5
- If $\mathbf{P} \neq \mathbf{NP}$, then there is no α-approximation algorithm for the general version of TSP for any polynomial time computable function α

Outline

1. Vertex cover
2. Traveling salesman
   - Metric TSP
   - Local search

LocalSearch

$s \leftarrow$ some initial solution
while there is a solution s' in the neighborhood of s which is better than s:
  $s \leftarrow s'$
return s

LocalSearch

```
s ← some initial solution
while there is a solution s' in the
neighborhood of s which is better than s:
  s ← s'
return s
```

- Computes a local optimum instead of a global optimum

LocalSearch

```
s ← some initial solution
while there is a solution s' in the
neighborhood of s which is better than s:
  s ← s'
return s
```

- Computes a local optimum instead of a global optimum
- The larger is the neighborhood, the better is the resulting solution and the higher is the running time

Local Search for TSP

- Let s and s' be two cycles visiting each vertex of the graph exactly once

Local Search for TSP

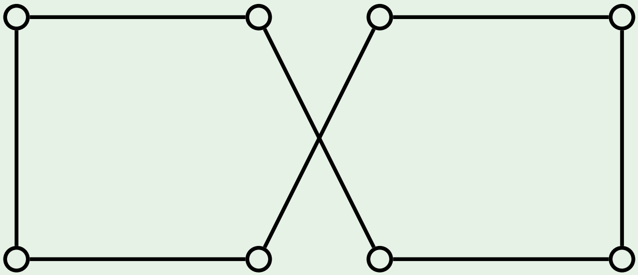
- Let s and s' be two cycles visiting each vertex of the graph exactly once
- The distance between s and s' is at most d, if one can get s' by deleting d edges from s and adding other d edges

Local Search for TSP

- Let s and s' be two cycles visiting each vertex of the graph exactly once
- The distance between s and s' is at most d, if one can get s' by deleting d edges from s and adding other d edges
- Neighborhood $N(s, r)$ with center s and radius r: all cycles with distance at most r from s

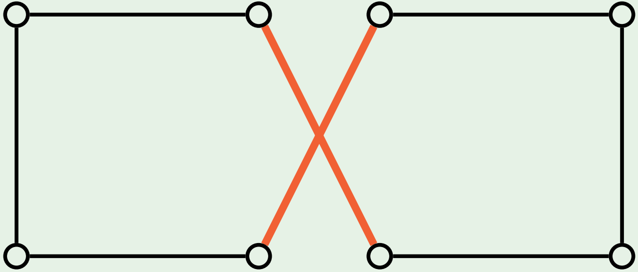
Example

Changing two edges in a suboptimal solution:



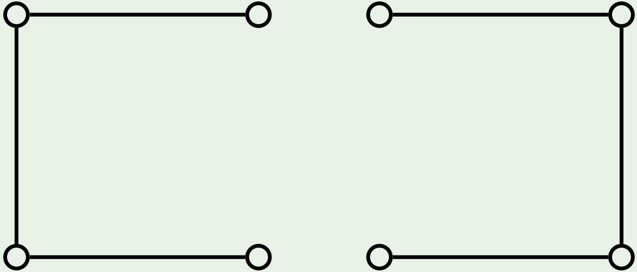
Example

Changing two edges in a suboptimal solution:



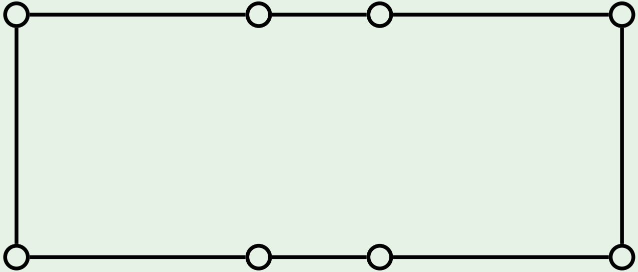
Example

Changing two edges in a suboptimal solution:



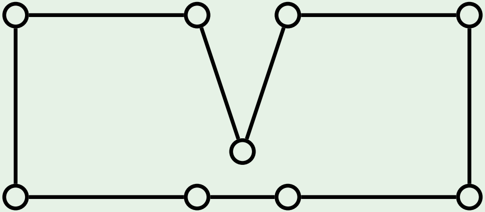
Example

Changing two edges in a suboptimal solution:



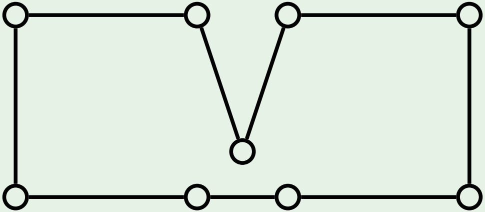
Example

A suboptimal solution that cannot be improved by changing two edges:



Example

A suboptimal solution that cannot be improved by changing two edges:



Need to allow changing three edges to improve this solution

Performance

- Trade-off between quality and running time of a single iteration

Performance

- Trade-off between quality and running time of a single iteration
- Still, the number of iterations may be exponential and the quality of the found cycle may be poor

Performance

- Trade-off between quality and running time of a single iteration
- Still, the number of iterations may be exponential and the quality of the found cycle may be poor
- But works well in practice

Coping with NP-completeness

- special cases
- intelligent exhaustive search
- approximation algorithms