TECHNISCHE UNIVERSITÄT DARMSTADT

The Traveling Salesman Problem

Outlook: Linear Programming

Obligatory Question

Versteht hier jemand kein Deutsch?

Informal Definition

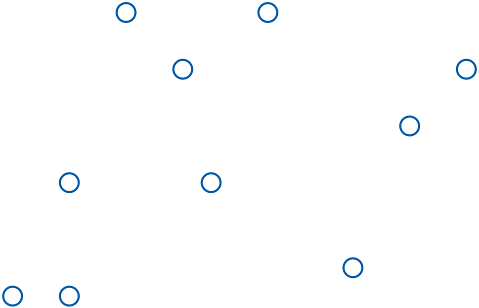
Traveling Salesman

Given a set of cities, and known distances between each pair of cities, find a tour that visits each city exactly once and that minimizes the total distance travelled.

The Traveling Salesman Problem

Example in the Euclidean Plane

TECHNISCHE UNIVERSITÄT DARMSTADT

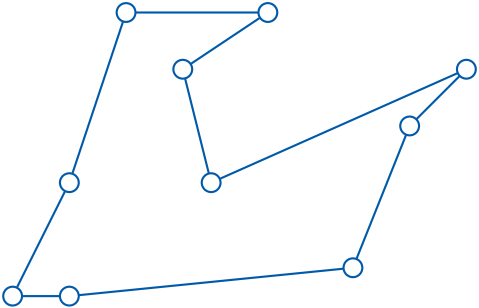


Given 10 cities and no obstacles . . .

The Traveling Salesman Problem

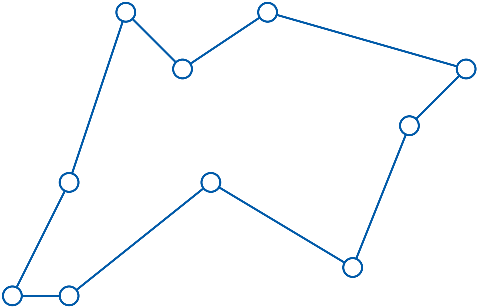
Example in the Euclidean Plane

TECHNISCHE UNIVERSITÄT DARMSTADT



This is a reasonable tour.

Example in the Euclidean Plane



This is the optimal tour.

Formal Definition

Traveling Salesman Function Problem

Given a complete undirected weighted graph $G = (V, E, c)$, find a Hamiltonian circuit of minimum total weight.

Traveling Salesman Decision Problem

Given an complete undirected weighted graph $G = (V, E, c)$ and a number x, decide whether there is a Hamiltonian circuit with total weight of at most x.

Other Variants

- If the input graph is not required to be complete, there might be no Hamilton cycle at all. If the problem is assumed to be feasible, we can compensate for missing edges by sufficiently long ones.

Other Variants

- If the input graph is not required to be complete, there might be no Hamilton cycle at all. If the problem is assumed to be feasible, we can compensate for missing edges by sufficiently long ones.
- If the input graph is not required to be undirected, the distance between two cities might depend on the travel direction. This problem is called the asymmetric TSP.

Other Variants

- If the input graph is not required to be complete, there might be no Hamilton cycle at all. If the problem is assumed to be feasible, we can compensate for missing edges by sufficiently long ones.
- If the input graph is not required to be undirected, the distance between two cities might depend on the travel direction. This problem is called the asymmetric TSP.
- If we search for a Hamiltonian circuit with the minimal weight of the weightiest edge, the problem is called the bottleneck TSP.

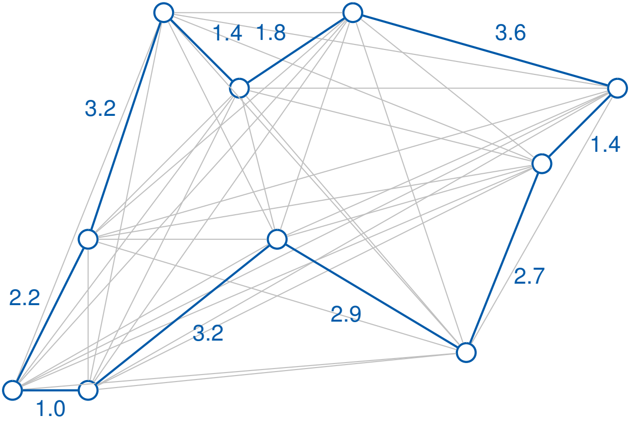
Other Variants

TECHNISCHE UNIVERSITÄT DARMSTADT

- If the input graph is not required to be complete, there might be no Hamilton cycle at all. If the problem is assumed to be feasible, we can compensate for missing edges by sufficiently long ones.
- If the input graph is not required to be undirected, the distance between two cities might depend on the travel direction. This problem is called the asymmetric TSP.
- If we search for a Hamiltonian circuit with the minimal weight of the weightiest edge, the problem is called the bottleneck TSP.
- Usually, the weights are assumed to be non-negative. If they also satisfy the triangle inequality, the problem is called the metric TSP. In particular, if vertices are identified with Cartesian coordinates in the Euclidean space, the problem is called the Euclidean TSP.

Example in the Euclidean Plane

TECHNISCHE UNIVERSITÄT DARMSTADT



Complexity

The Function Problem is

NP-hard

The Decision Problem is

NP-complete

What is a Turing Machine?

Turing Machine

Worst-case Runtime

Let A be a (deterministic) algorithm. Its time complexity $T_A(n)$ is the maximum amount of time taken on any input of size n.

Algorithm Complexity

Worst-case Runtime

Let A be a (deterministic) algorithm. Its time complexity $T_A(n)$ is the maximum amount of time taken on any input of size n.

An algorithm A is called a

- constant time algorithm, if: $T_A(n) \in O(1)$
- linear time algorithm, if: $T_A(n) \in O(n)$
- polynomial time algorithm, if: $\exists k : T_A(n) \in O(n^k)$
- exponential time algorithm, if: $\exists k : T_A(n) \in O(2^{n^k})$

Algorithm Complexity

Worst-case Runtime

Let A be a (deterministic) algorithm. Its time complexity $T_A(n)$ is the maximum amount of time taken on any input of size n.

An algorithm A is called a

- constant time algorithm, if: $T_A(n) \in O(1)$
- linear time algorithm, if: $T_A(n) \in O(n)$
- polynomial time algorithm, if: $\exists k : T_A(n) \in O(n^k)$
- exponential time algorithm, if: $\exists k : T_A(n) \in O(2^{n^k})$

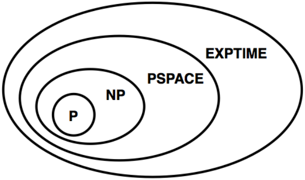
The same definitions hold for the space complexity $S_A(n)$.

Problem Complexity

Complexity Classes

- P (PTIME) is the class of all decision problems which can be solved in polynomial time.
- NP is the class of all decision problems whose solutions can be verified in polynomial time.
- PSPACE is the class of all decision problems which can be solved in polynomial space.
- EXPTIME is the class of all decision problems which can be solved in exponential time.

Assumed Inclusion



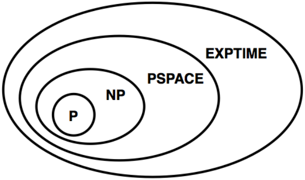
Problem Complexity

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Complexity Classes

- P (PTIME) is the class of all decision problems which can be solved in polynomial time.
- NP is the class of all decision problems whose solutions can be verified in polynomial time.
- PSPACE is the class of all decision problems which can be solved in polynomial space.
- EXPTIME is the class of all decision problems which can be solved in exponential time.

Assumed Inclusion



Hardness and Completeness

A problem p is called CLASS-hard, if there is a polynomial time reduction from all problems in CLASS to p. A problem is called CLASS-complete, if it is CLASS-hard and in CLASS.

TECHNISCHE UNIVERSITÄT DARMSTADT

The Function Problem is

NP-hard

The Decision Problem is

NP-complete

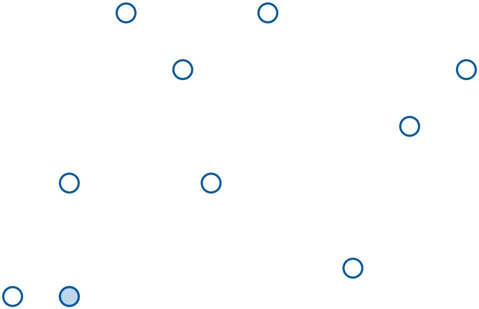
. . . even with Euclidean distances!

Constructive Heuristics

Find good solutions in polynomial time!

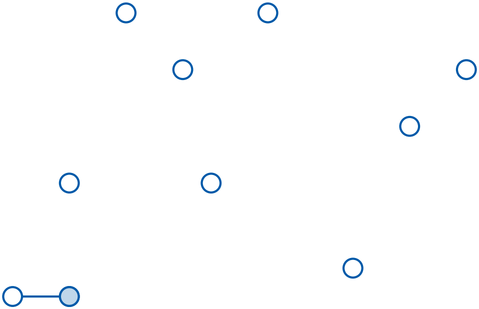
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



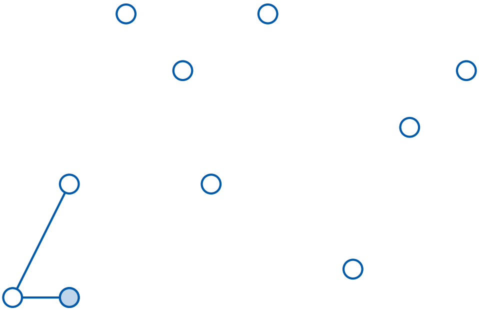
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



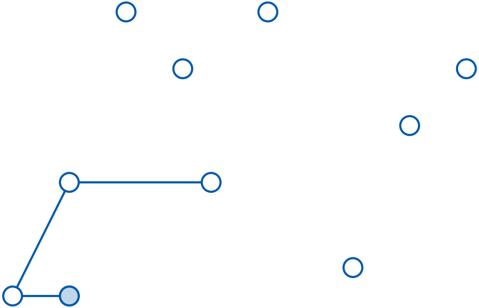
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



Nearest Neighbour algorithm

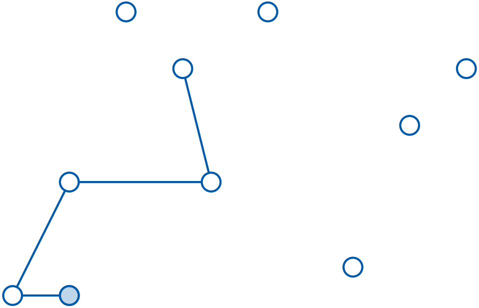
Successively visit the nearest unvisited city.



TECHNISCHE UNIVERSITÄT DARMSTADT

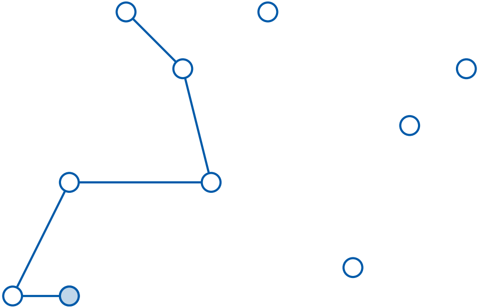
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



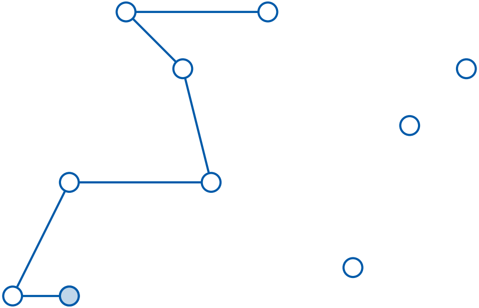
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



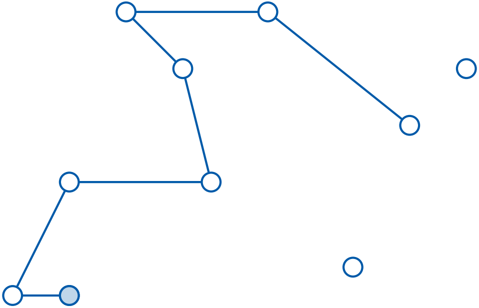
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



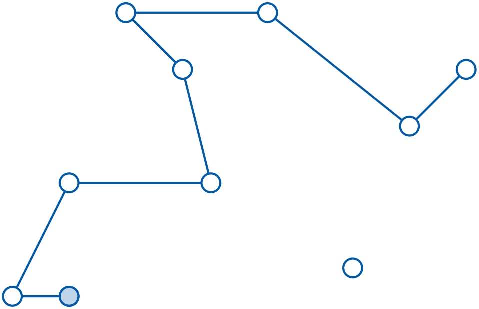
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



Nearest Neighbour algorithm

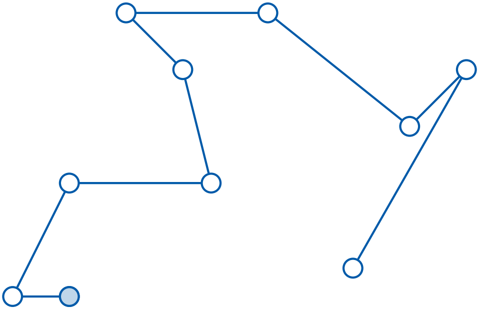
Successively visit the nearest unvisited city.



TECHNISCHE UNIVERSITÄT DARMSTADT

Nearest Neighbour algorithm

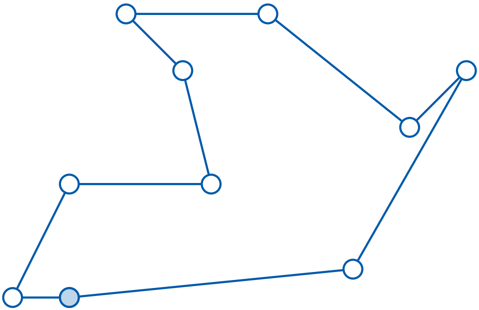
Successively visit the nearest unvisited city.



TECHNISCHE UNIVERSITÄT DARMSTADT

Nearest Neighbour algorithm

Successively visit the nearest unvisited city.



Nearest Neighbour algorithm

Successively visit the nearest unvisited city.

- The NN algorithm is easy to implement and runs in $O(V^2)$.

Nearest Neighbour algorithm

Successively visit the nearest unvisited city.

- The NN algorithm is easy to implement and runs in $O(V^2)$.
- For randomly distributed cities in the plane, the NN algorithm on average finds a tour which is approximately 25 % longer than the optimal tour.

Nearest Neighbour algorithm

Successively visit the nearest unvisited city.

- The NN algorithm is easy to implement and runs in $O(V^2)$.
- For randomly distributed cities in the plane, the NN algorithm on average finds a tour which is approximately 25 % longer than the optimal tour.
- The NN algorithm may not find any feasible tour at all.

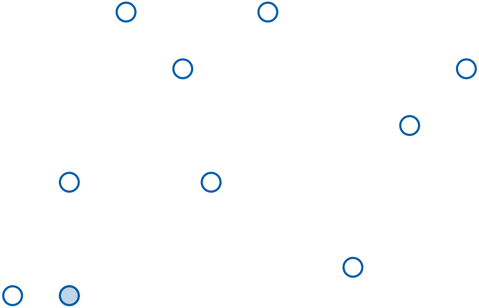
Nearest Neighbour algorithm

Successively visit the nearest unvisited city.

- The NN algorithm is easy to implement and runs in $O(V^2)$.
- For randomly distributed cities in the plane, the NN algorithm on average finds a tour which is approximately 25 % longer than the optimal tour.
- The NN algorithm may not find any feasible tour at all.
- It is easy to construct distances for any given number of cities where the NN algorithm finds the unique worst of all possible tours.

Minimum Spanning Tree algorithm

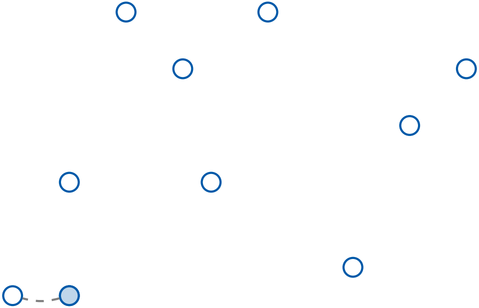
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

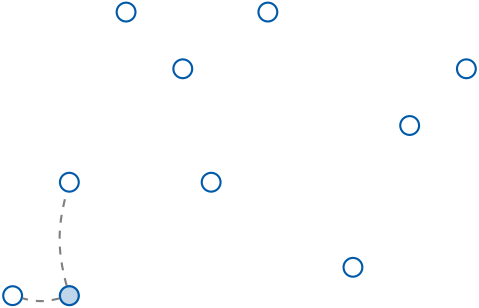
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

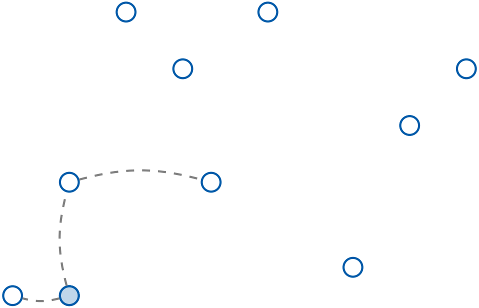
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

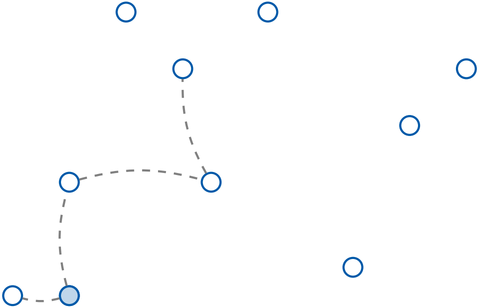
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



TECHNISCHE UNIVERSITÄT DARMSTADT

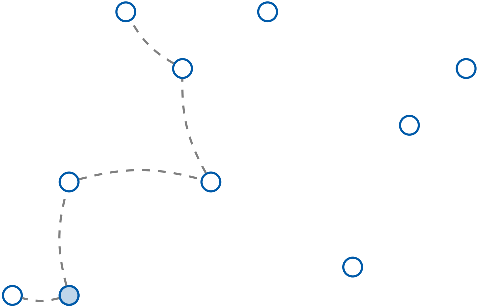
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



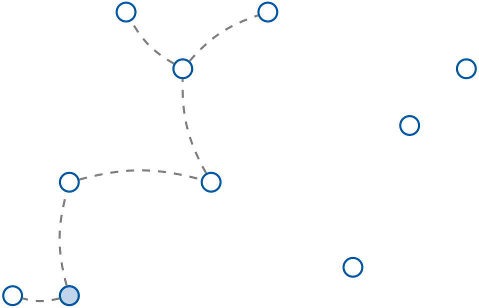
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim’s algorithm.



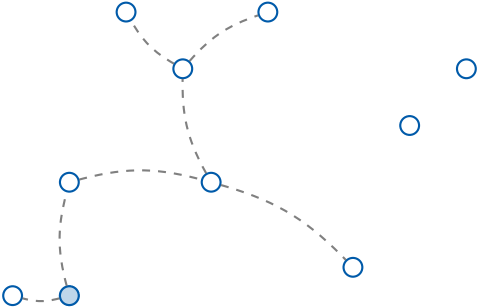
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



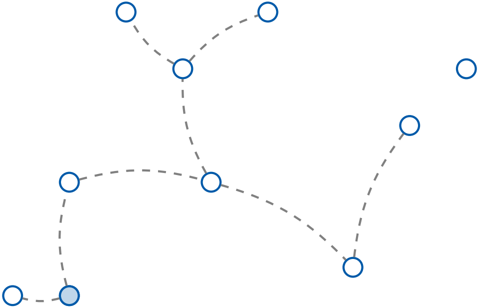
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



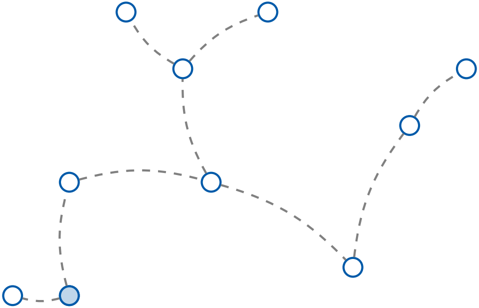
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.



Minimum Spanning Tree algorithm

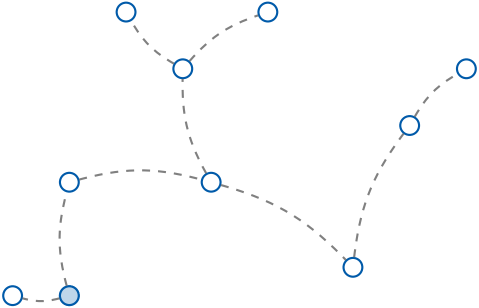
Step 1: Construct a minimal spanning tree, e.g. with Prim’s algorithm.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

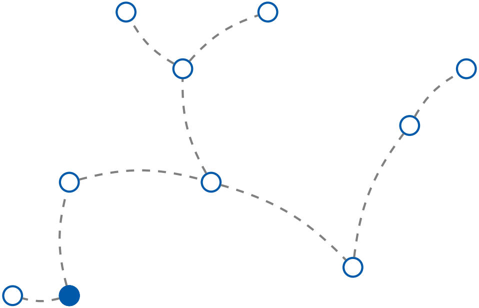
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

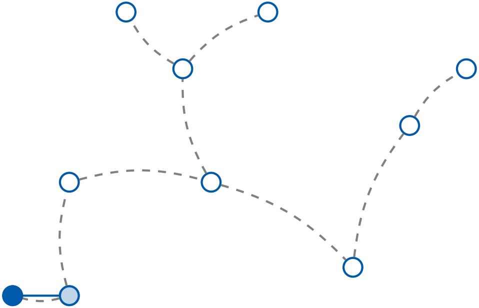
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.

Step 2: Traverse the MST by DFS, but skip already visited cities.



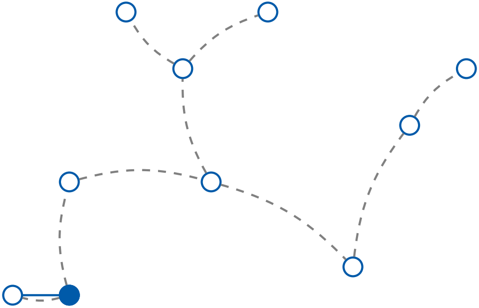
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim’s algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

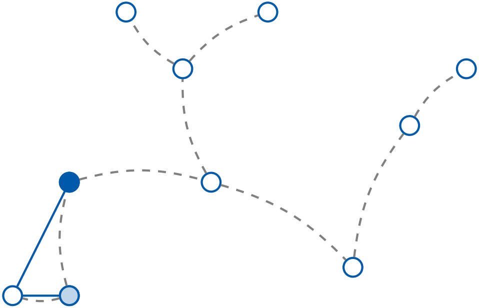
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

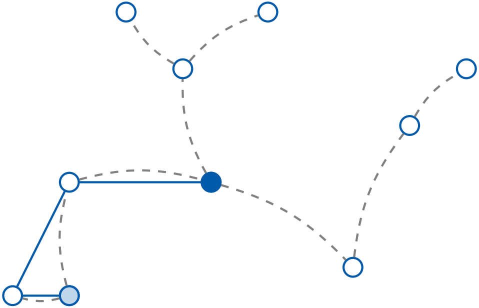
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.

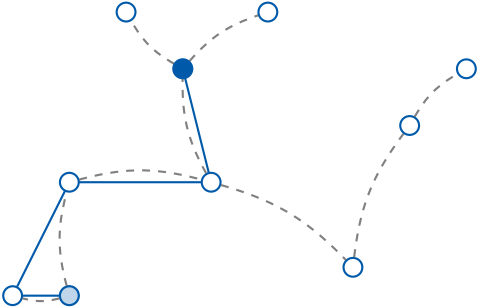


TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

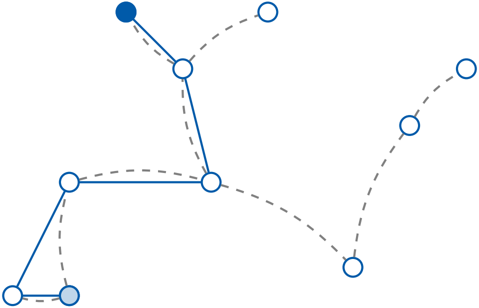
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.

Step 2: Traverse the MST by DFS, but skip already visited cities.



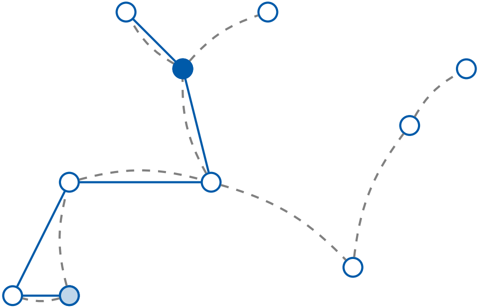
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

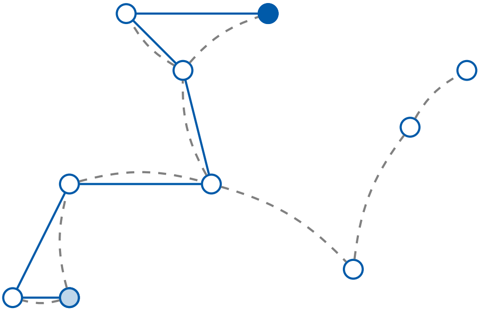
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

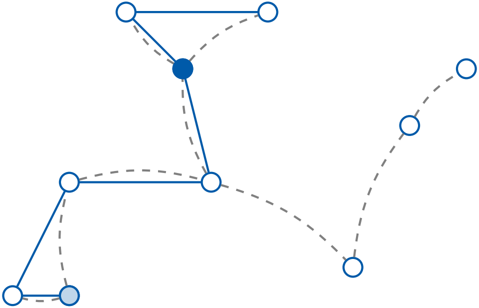
Step 1: Construct a minimal spanning tree, e.g. with Prim’s algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

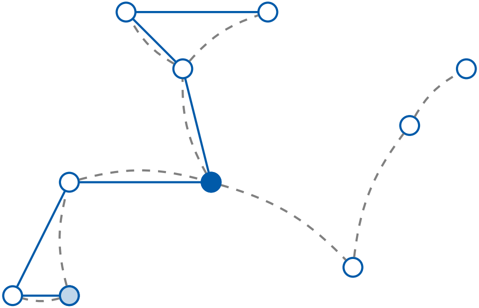
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

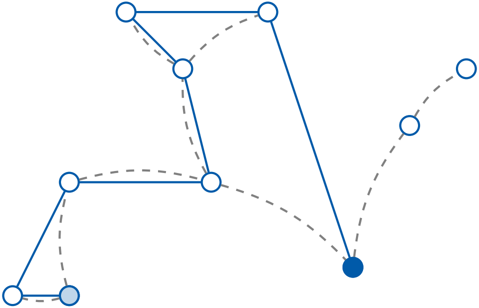
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



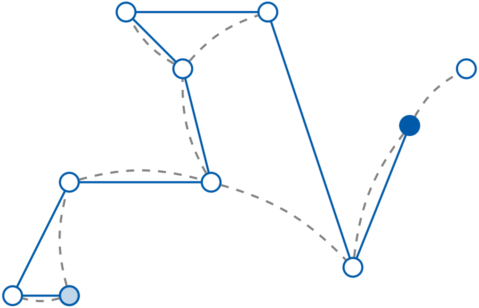
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

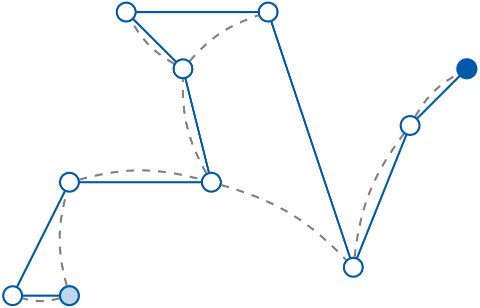
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

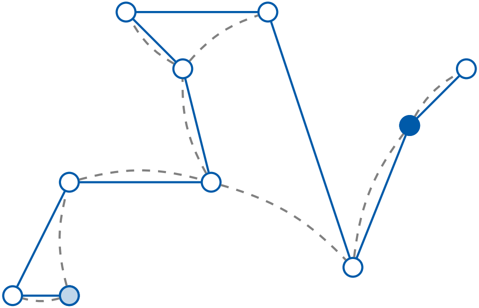
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.

Step 2: Traverse the MST by DFS, but skip already visited cities.



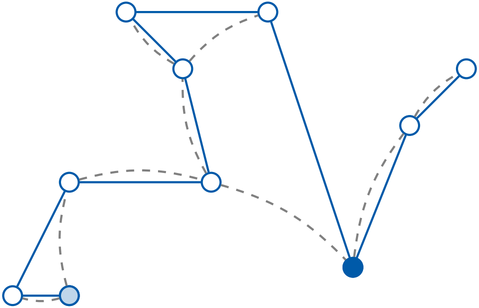
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.

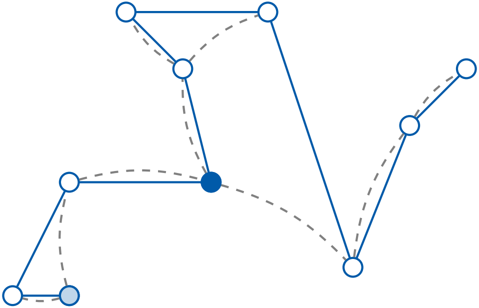


TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim’s algorithm.

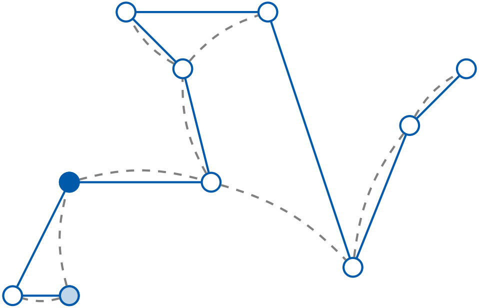
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

Minimum Spanning Tree algorithm

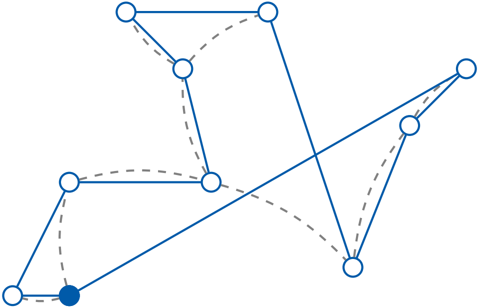
Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



TECHNISCHE UNIVERSITÄT DARMSTADT

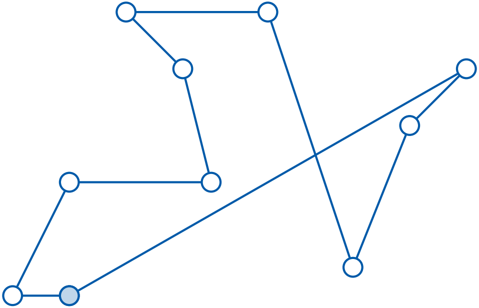
Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim’s algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.



Double Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.

- The DMST algorithm runs in $O(V^2 \log V)$.

Double Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.

- The DMST algorithm runs in $O(V^2 \log V)$.
- In any graph, the weight of a MST is less than the weight of the optimal tour. Therefore, if the triangle inequality holds, the constructed tour is less than twice as long as the optimal tour. (approximation ratio 2)

Double Minimum Spanning Tree algorithm

Step 1: Construct a minimal spanning tree, e.g. with Prim's algorithm.
Step 2: Traverse the MST by DFS, but skip already visited cities.

- The DMST algorithm runs in $O(V^2 \log V)$.
- In any graph, the weight of a MST is less than the weight of the optimal tour. Therefore, if the triangle inequality holds, the constructed tour is less than twice as long as the optimal tour. (approximation ratio 2)
- A variant of the DMST algorithm, the Christofides algorithm, achieves an approximation ratio of 1.5 in $O(V^3)$.

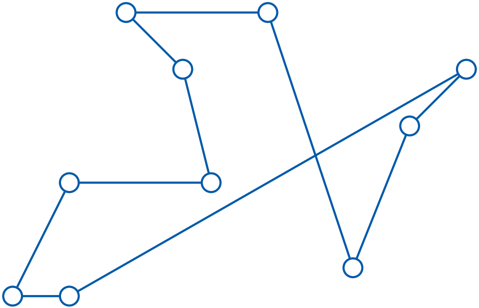
Improvement Heuristics

Construct better solutions from existing ones!

Pairwise Exchange (2-opt) algorithm

Step 1: Remove two disjoint edges from the tour.

Step 2: Reconnect both paths to a valid tour (as short as possible).

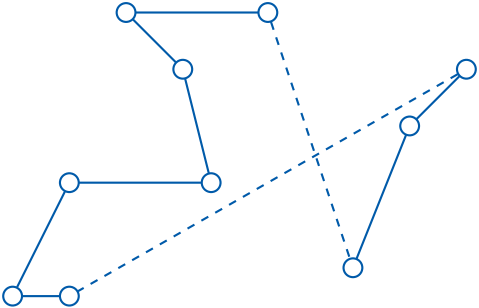


TECHNISCHE UNIVERSITÄT DARMSTADT

Pairwise Exchange (2-opt) algorithm

Step 1: Remove two disjoint edges from the tour.

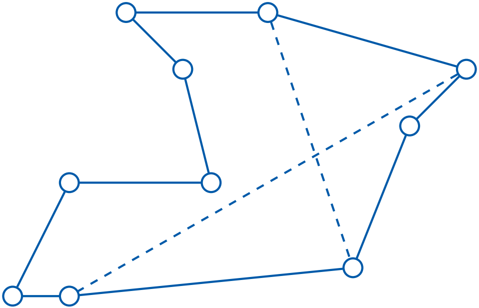
Step 2: Reconnect both paths to a valid tour (as short as possible).



Pairwise Exchange (2-opt) algorithm

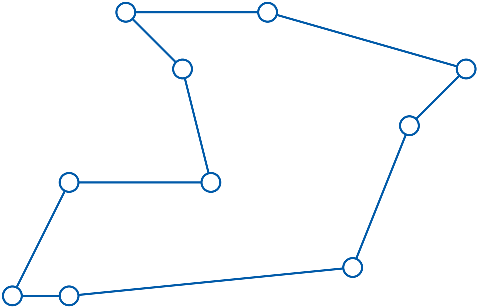
Step 1: Remove two disjoint edges from the tour.

Step 2: Reconnect both paths to a valid tour (as short as possible).



Pairwise Exchange (2-opt) algorithm

Step 1: Remove two disjoint edges from the tour.
Step 2: Reconnect both paths to a valid tour (as short as possible).



Lin-Kernighan (variable-opt) algorithm

Step 1: Choose a suitable *k* for the k-opt algorithm.
Step 2: Remove *k* edges from the tour.
Step 3: Reconnect the paths to a valid tour (as short as possible).

Improvement Heuristics

Lin-Kernighan (variable-opt) algorithm

Step 1: Choose a suitable k for the k-opt algorithm.
Step 2: Remove k edges from the tour.
Step 3: Reconnect the paths to a valid tour (as short as possible).

- The LK algorithm (a single iteration) runs in approximately $O(V^{2.2})$.

Improvement Heuristics

Lin-Kernighan (variable-opt) algorithm

Step 1: Choose a suitable *k* for the k-opt algorithm.
Step 2: Remove *k* edges from the tour.
Step 3: Reconnect the paths to a valid tour (as short as possible).

- The LK algorithm (a single iteration) runs in approximately $O(V^{2.2})$.
- Even in Euclidean TSPs, 2-opt can take an exponential number of steps. In probabilistic instances, the expected number of steps is polynomial.

Improvement Heuristics

Lin-Kernighan (variable-opt) algorithm

Step 1: Choose a suitable *k* for the k-opt algorithm.
Step 2: Remove *k* edges from the tour.
Step 3: Reconnect the paths to a valid tour (as short as possible).

- The LK algorithm (a single iteration) runs in approximately $O(V^{2.2})$.
- Even in Euclidean TSPs, 2-opt can take an exponential number of steps. In probabilistic instances, the expected number of steps is polynomial.
- There is no guaranteed improvement in tour length. In probabilistic instances, 2-opt approximately achieves a 5% gap, 3-opt a 3% gap.

Improvement Heuristics

Lin-Kernighan (variable-opt) algorithm

Step 1: Choose a suitable k for the k-opt algorithm.
Step 2: Remove k edges from the tour.
Step 3: Reconnect the paths to a valid tour (as short as possible).

- The LK algorithm (a single iteration) runs in approximately $O(V^{2.2})$.
- Even in Euclidean TSPs, 2-opt can take an exponential number of steps. In probabilistic instances, the expected number of steps is polynomial.
- There is no guaranteed improvement in tour length. In probabilistic instances, 2-opt approximately achieves a 5% gap, 3-opt a 3% gap.
- Lin-Kernighan-Johnson can solve many instances to optimality.

Ant Colony Optimization algorithm

Send out a large number of virtual ants to explore many possible tours. As a simple method of communication, these ants rate edges by means of virtual pheromones. Each individual ant chooses its next destination randomized, based on a heuristic weighting of several simple factors:

Ant Colony Optimization algorithm

Send out a large number of virtual ants to explore many possible tours. As a simple method of communication, these ants rate edges by means of virtual pheromones. Each individual ant chooses its next destination randomized, based on a heuristic weighting of several simple factors:

- Near (visible) cities have a higher chance of being chosen.
- Edges with much pheromone have a higher chance of being chosen.
- Ants which completed a tour deposit pheromone on all edges traversed. The shorter the tour, the more pheromone is deposited.
- Over time, pheromone trails evaporate.

Exact Algorithms

Find a guaranteed optimal solution!

Exact Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

Brute Force algorithm

Try all permutations of cities and remember which one is cheapest.

Brute Force algorithm

Try all permutations of cities and remember which one is cheapest.

- The BF algorithm runs in $O(V!)$. Examples:

# cities	# tours	est. time
5	12	$12\mu s$
10	181 000	$0.2s$
15	87×10^9	$12h$
20	60×10^{15}	$2000y$
n	$(n-1)!/2$	...

- Impractical for real-world instances.

TECHNISCHE UNIVERSITÄT DARMSTADT

Dynamic Programming algorithm

Solve the shortest subtour problem on subsequent larger subgraphs:
If we are at city i and still have to visit all cities in S, then we have

$$c^*(i, S) = \min_{j \in S} \{ c(i, j) + c^*(j, S \setminus \{j\}) \}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Dynamic Programming algorithm

Solve the shortest subtour problem on subsequent larger subgraphs:
If we are at city i and still have to visit all cities in S, then we have

$$c^*(i, S) = \min_{j \in S}\{c(i, j) + c^*(j, S \setminus \{j\})\}$$

- The DP algorithm runs in $O(n^2 \cdot 2^n)$, but requires exponential space.

TECHNISCHE UNIVERSITÄT DARMSTADT

Dynamic Programming algorithm

Solve the shortest subtour problem on subsequent larger subgraphs:
If we are at city i and still have to visit all cities in S, then we have

$$c^*(i, S) = \min_{j \in S} \{ c(i, j) + c^*(j, S \setminus \{j\}) \}$$

- The DP algorithm runs in $O(n^2 \cdot 2^n)$, but requires exponential space.
- It can be modified to only need polynomial space, at the expense of time complexity (e.g. $O(poly(n) \cdot 2^n)$ or $O(4^n)$).

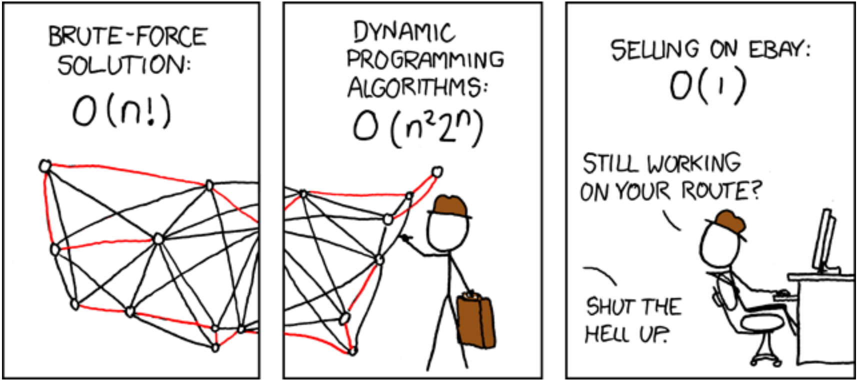
Dynamic Programming algorithm

Solve the shortest subtour problem on subsequent larger subgraphs:
If we are at city i and still have to visit all cities in S, then we have

$$c^*(i, S) = \min_{j \in S}\{c(i, j) + c^*(j, S \setminus \{j\})\}$$

- The DP algorithm runs in $O(n^2 \cdot 2^n)$, but requires exponential space.
- It can be modified to only need polynomial space, at the expense of time complexity (e.g. $O(poly(n) \cdot 2^n)$ or $O(4^n)$).
- It is an open problem if an algorithm with a base less than 2, e.g. with runtime in $O(poly(n) \cdot 1.999^n)$, exists.

Summary



http://xkcd.com/399/

TECHNISCHE UNIVERSITÄT DARMSTADT

What comes after ADM?

Introduction

We have seen many algorithms to various problems.

Introduction

We have seen many algorithms to various problems.

Is there one algorithm to rule them all?

Of course

All problems in P can be reduced to all P-hard problems.
All problems in NP can be reduced to all NP-hard problems.

Introduction

The right question is:
Which problem makes for easy reductions?

Linear constraints are intuitive for humans.

Definition

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b\}$$

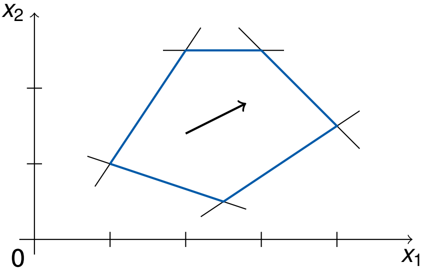
Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b\}$$



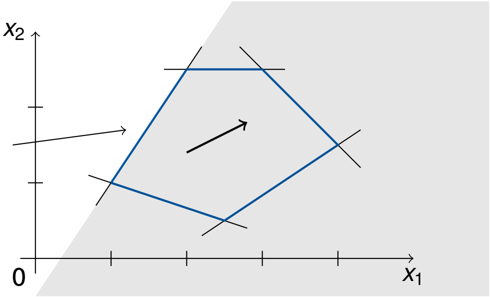
TECHNISCHE UNIVERSITÄT DARMSTADT

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \le b\}$$

$1.5 \cdot x_1 - x_2 \le 0.5$

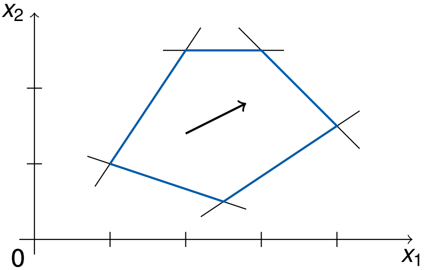


Definition

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

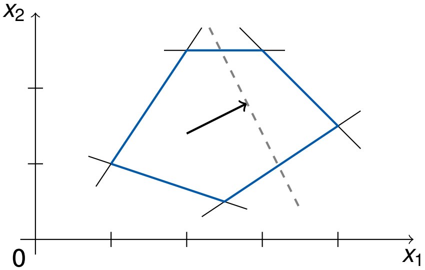
$$\min \{c^T x \mid Ax \leq b\}$$



Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b\}$$

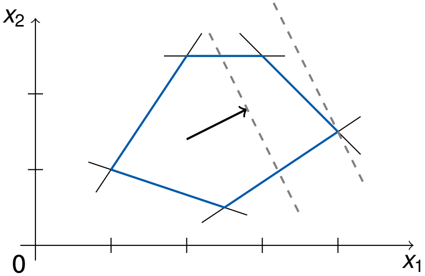


Definition

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b\}$$

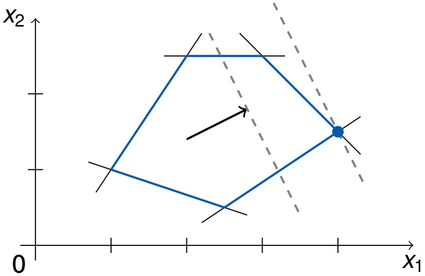


Definition

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b\}$$



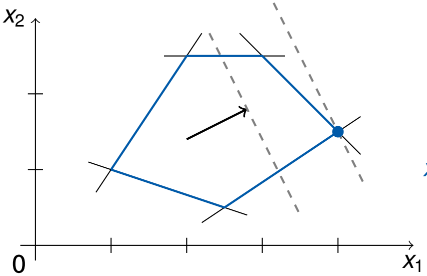
Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

Linear Program (LP)

Optimize a linear objective function over a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b\}$$



$c^T x^* = 3.8$

$x^* = (4, 1.5)$

Examples

Pottery

A potter is making cups and plates. It takes her 6 minutes to make a cup and 3 minutes to make a plate. Each cup uses 3/4 lb. of clay and each plate uses one lb. of clay. She has 20 hours available for making the cups and plates and has 250 lbs. of clay on hand. She makes a profit of $2 on each cup and $1.50 on each plate. How many cups and how many plates should she make in order to maximize her profit?

Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

Pottery

A potter is making cups and plates. It takes her 6 minutes to make a cup and 3 minutes to make a plate. Each cup uses 3/4 lb. of clay and each plate uses one lb. of clay. She has 20 hours available for making the cups and plates and has 250 lbs. of clay on hand. She makes a profit of $2 on each cup and $1.50 on each plate. How many cups and how many plates should she make in order to maximize her profit?

maximize

$$2x + 1.5y$$

subject to

$$\begin{aligned} 6x + 3y &\leq 20 \cdot 60 \qquad & x &\geq 0 \\ 0.75x + y &\leq 250 \qquad & y &\geq 0 \end{aligned}$$

Examples

Max Flow

Given a directed weighted graph $G = (V, E, c)$ with $c > 0$ and two distinguished nodes s and t, maximize the network flow from s to t.

Max Flow

Given a directed weighted graph $G = (V, E, c)$ with $c > 0$ and two distinguished nodes s and t, maximize the network flow from s to t.

maximize

$$\sum_{(s,f)\in E} x_{(s,f)}$$

subject to

$$\forall v \in V \setminus \{s, t\} : \quad \sum_{(i,v)\in E} x_{(i,v)} = \sum_{(v,o)\in E} x_{(v,o)}$$

$$\forall e \in E : \quad 0 \leq x_e \leq c_e$$

Shortest Path

Given a directed weighted graph $G = (V, E, c)$ without a negative-weight cycle and two distinguished nodes s and t, find the shortest path from s to t.

Shortest Path

Given a directed weighted graph $G = (V, E, c)$ without a negative-weight cycle and two distinguished nodes s and t, find the shortest path from s to t.

maximize

$$x_t$$

subject to

$$x_s = 0$$

$$\forall (u, v) \in E : \quad x_v \leq x_u + c_{(u,v)}$$

Discussion

What does the term Programming mean?

Discussion

What does the term Programming mean?

Traditional Programming

- How can the solution be found?
- “Calculate a space curve for the 3D printer’s laser to follow.”

Discussion

What does the term Programming mean?

Traditional Programming

- How can the solution be found?
- “Calculate a space curve for the 3D printer’s laser to follow.”

Linear Programming

- What does a solution look like?
- “Load a CAD model of the desired object onto the 3D printer.”

Algorithms

Simplex Method

If a feasible solution exists and if the objective function is bounded, the optimal objective value is attained at a vertex. Start at any vertex, then successively follow any edge to a better vertex until there is none.

Interior Point Method

Start at any feasible point in the polyhedron, then derive any other point which is still in the polyhedron and has a better objective value. Do this fast enough and find a suitable termination criterion.

Generalization

TECHNISCHE UNIVERSITÄT DARMSTADT

The LP Decision Problem is (weakly) P-complete.

Is there a similar modeling language for NP?

Generalization

Integer Linear Program (ILP)

Optimize a linear objective function over the integer points in a convex polyhedron:

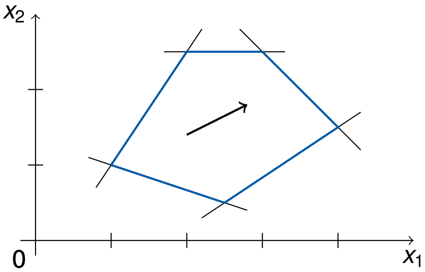
$$\min \{c^T x \mid Ax \leq b, x \in \mathbb{Z}^n\}$$

Generalization

Integer Linear Program (ILP)

Optimize a linear objective function over the integer points in a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b, x \in \mathbb{Z}^n\}$$

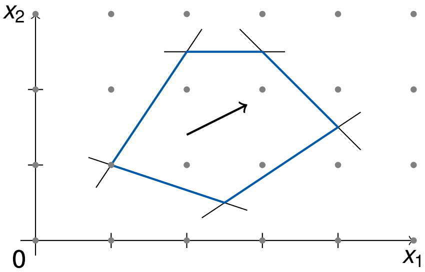


Generalization

Integer Linear Program (ILP)

Optimize a linear objective function over the integer points in a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b, x \in \mathbb{Z}^n\}$$

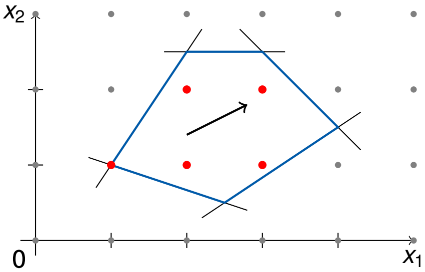


Generalization

Integer Linear Program (ILP)

Optimize a linear objective function over the integer points in a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b, x \in \mathbb{Z}^n\}$$

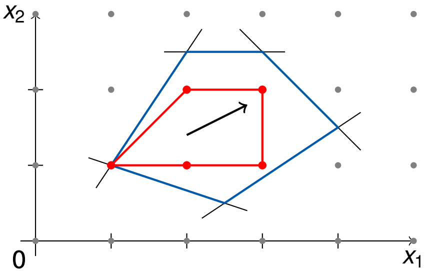


Generalization

Integer Linear Program (ILP)

Optimize a linear objective function over the integer points in a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b, x \in \mathbb{Z}^n\}$$



Knapsack Problem

Given a set of items I, each with a size S and a value V.
Maximize the total value of a knapsack with size K.

Knapsack Problem

Given a set of items I, each with a size S and a value V.
Maximize the total value of a knapsack with size K.

maximize

$$\sum_{i \in I} V_i x_i$$

subject to

$$\sum_{i \in I} S_i x_i \leq K$$

$$\forall i \in I : \quad x_i \in \{0, 1\}$$

Examples

Traveling Salesman Problem

Given a complete undirected weighted graph $G = (V, E, c)$, find a Hamiltonian circuit of minimum total weight.

Traveling Salesman Problem

Given a complete undirected weighted graph $G = (V, E, c)$, find a Hamiltonian circuit of minimum total weight.

minimize

$$\sum_{(i,j)\in E} c_{(i,j)} x_{(i,j)}$$

subject to

$$\forall j \in V: \quad \sum_{i \in V} x_{(i,j)} = 1 \qquad \forall i \in V: \quad \sum_{j \in V} x_{(i,j)} = 1$$

$$\forall S \subsetneq V, |S| > 1: \quad \sum_{i \in S, j \in S, i \neq j} x_{(i,j)} \leq |S| - 1$$

Generalization

Mixed-Integer Linear Program (MILP)

Optimize a linear objective function over integer hyperspaces in a convex polyhedron:

$$\min \{c^T x \mid Ax \leq b, x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}\}$$

The best of both worlds, and more:

- semi-continuous variables, discontinuous domains, ...
- logic constraints: if-then, either-or, if-and-only-if, ...
- piecewise linearization, variable-product elimination, ...

Algorithms

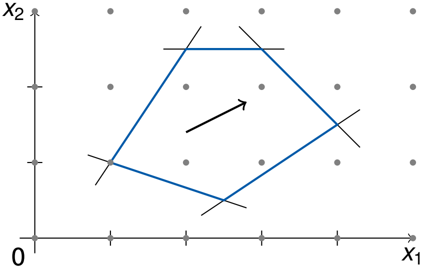
Branch and Bound method

Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.

Algorithms

Branch and Bound method

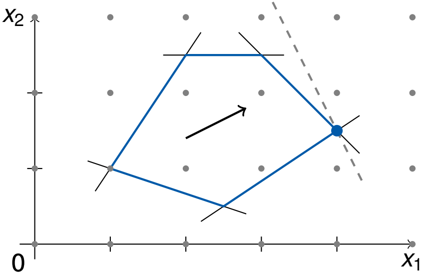
Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and Bound method

Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.

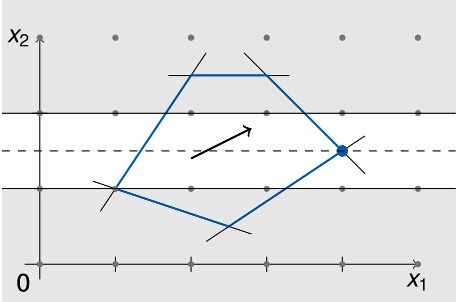


$c^T x^* = 3.8$

$x^* = (4, 1.5)$

Branch and Bound method

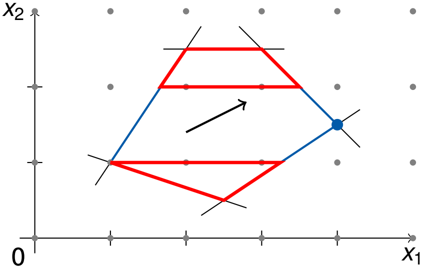
Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and Bound method

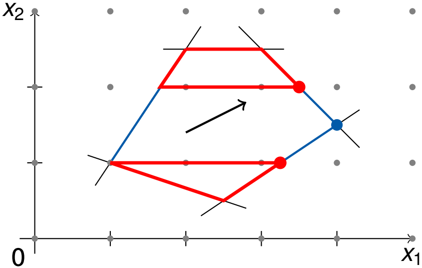
Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



Algorithms

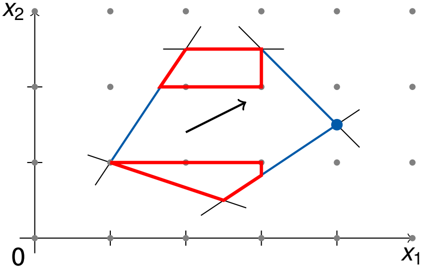
Branch and Bound method

Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



Branch and Bound method

Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.

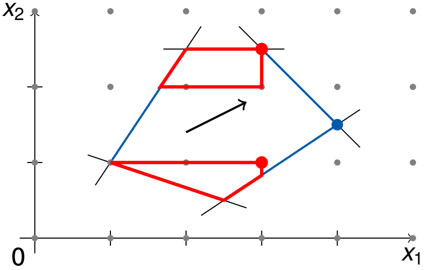


Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and Bound method

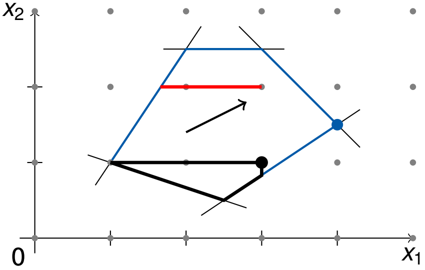
Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and Bound method

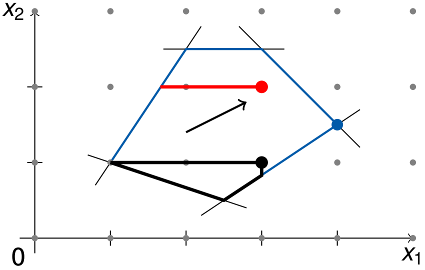
Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



TECHNISCHE UNIVERSITÄT DARMSTADT

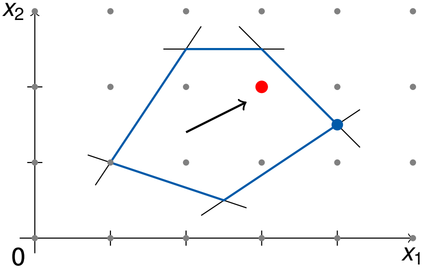
Branch and Bound method

Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



Branch and Bound method

Solve the LP-Relaxation, i.e., assume all variables to be continuous. If the solution does not satisfy the integer constraints, branch and repeat with the subproblems.



Generalization

The ILP/MILP Decision Problem is NP-complete.

Is there a similar modeling language for PSPACE?

Generalization

Quantified Mixed-Integer Linear Program (QMILP)

Optimize a linear objective function over integer hyperspaces in a convex polyhedron with quantified variables:

$$\min \{c^T x \mid Q(x) : Ax \leq b, x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}\}$$

Quantifiers are known from logic, e.g.:

$$Q(x) = \exists\, x_1 \;\forall\, x_2 \;\exists\, x_3, x_4 \;Я\, x_5 \;\exists\, x_6 \;\ldots$$

$\exists$ = exists $\forall$ = for all $Я$ = for random

Generalization

What consequences does the quantification have?

Generalization

What consequences does the quantification have?

- The meaning of the objective function changes:

$$\min c^T x \;\tilde{=}\; \min_{x_1} \max_{x_2} \min_{x_3, x_4} \mathop{\mathrm{E}}_{x_5} \min_{x_6} \; c^T x$$

Generalization

What consequences does the quantification have?

- The meaning of the objective function changes:

$$\min c^T x \ \tilde{=}\ \min_{x_1} \max_{x_2} \min_{x_3, x_4} \underset{x_5}{\mathrm{E}} \min_{x_6} \ c^T x$$

- The problem shows tendencies of a two-person game:
  - The existential player wants to stay in the polyhedron and minimize the objective function.
  - The universal player wants to leave the polyhedron and maximize the objective function.

Examples

Single-player Games (Patience games)

Kondike, Freecell, . . .

Examples

Single-player Games (Patience games)

Kondike, Freecell, . . .

Two-player Games (Stategy games)

Tic-Tac-Toe, Chess, Checkers, Go, Gomoku, Reversi, . . .

Examples

Single-player Games (Patience games)

Kondike, Freecell, . . .

Two-player Games (Stategy games)

Tic-Tac-Toe, Chess, Checkers, Go, Gomoku, Reversi, . . .

Two-player Games with Chance

Ludo, Backgammon, . . .

Production Planning under Uncertainty

- What is the most profitable production strategy for a given customer demand?

Production Planning under Uncertainty

- What is the most profitable production strategy for a given customer demand?
- What is the best investment decision, if the customer demand is uncertain?

Examples

Production Planning under Uncertainty

- What is the most profitable production strategy for a given customer demand?
- What is the best investment decision, if the customer demand is uncertain?

Booster Stations under Uncertainty

- What is the most efficient pump operation for a given load collective?

Examples

Production Planning under Uncertainty

- What is the most profitable production strategy for a given customer demand?
- What is the best investment decision, if the customer demand is uncertain?

Booster Stations under Uncertainty

- What is the most efficient pump operation for a given load collective?
- What is the best initial topology, if the load collective is uncertain?

Software

Want to try it out yourself?

`http://gusek.sourceforge.net/`