Generalization of Algorithms for the Orienteering Problem

Etienne Palanga

January 20, 2023

2023-01-20

Generalization of Algorithms for the Orienteering Problem

Generalization of Algorithms for the Orienteering Problem

Etienne Palanga

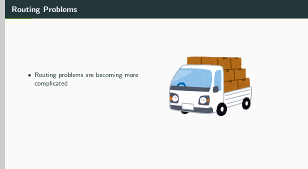
January 20, 2023

Introduction

Introduction

Routing Problems

- Routing problems are becoming more complicated



- Routing problems not only more prevalent and more complicated
  - Navigation of vehicles or people between a predetermined set of points (like a set of customers)
  - Some constraint or goal
  - $\Rightarrow$ Want optimal path or paths
- Navigation in general
  - Want to save time/fuel (shortest path)
  - Want a beautiful trail to walk (highest quality path)
- Online Shopping $\rightarrow$ shipping packages to customers
  - Want to deliver the most packages in the smallest amount of time

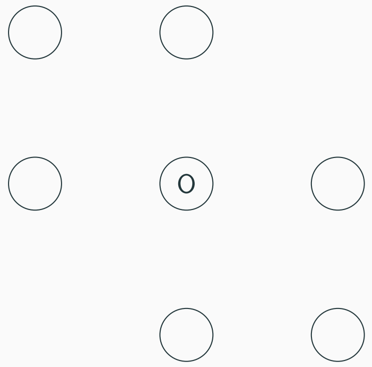
Routing Problems

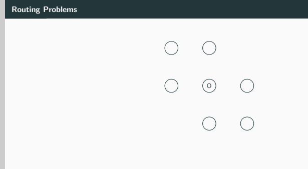
- Routing problems are becoming more complicated
  - Amazon with ≈ 4.75 billion shipped packages [8]

Routing Problems
- Routing problems are becoming more complicated
  - Amazon with ≈ 4.75 billion shipped packages [8]

- Example: Amazon
  - 4.75 billion shipped packages
  - Need a way to manage this
- Frequent Task: Optimize Routes in some respect
  - Reduce fuel consumption/travel time
  - Maximize profit gained
  - How do I deliver the most goods while on a time budget?
  - ... (you can think of a lot of things)

Routing Problems

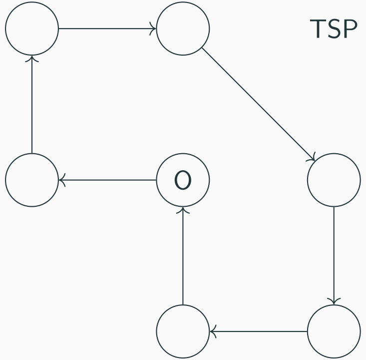


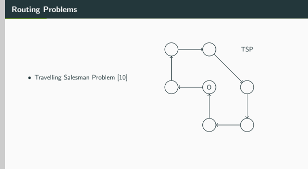


- Such problems are no stranger to computer science

Routing Problems

- Travelling Salesman Problem [10]

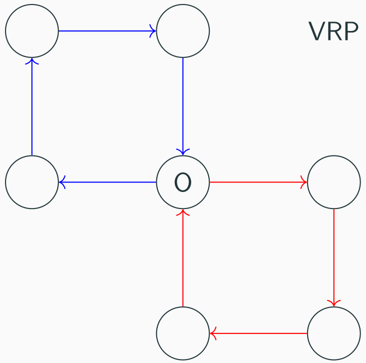


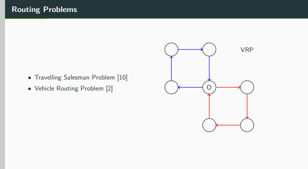


- Most famous: Travelling Salesman Problem
  - Graph
  - Visit all nodes exactly once and return to the origin using the least distance
- But there are also others

Routing Problems

- Travelling Salesman Problem [10]
- Vehicle Routing Problem [2]

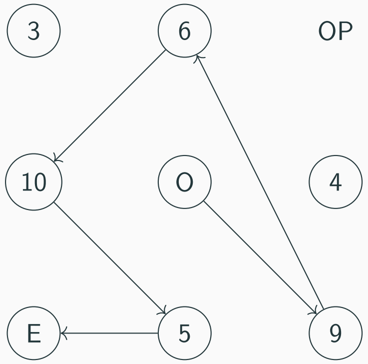


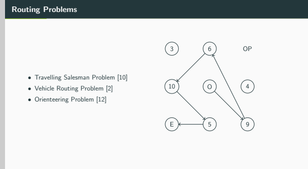


- Vehicle Routing Problem
  - Generalization of the TSP
  - Multiple vehicles starting at the origin
  - ⇒ multiple routes minimizing the travel cost

Routing Problems

- Travelling Salesman Problem [10]
- Vehicle Routing Problem [2]
- Orienteering Problem [12]





- **Orienteering Problem**
  - No need to visit all nodes
  - Nodes have scores/profit, edges have weights (omitted here for clarity)
  - Time/distance limit
  - Find a path (not necessarily cycle) with maximum profit that does not violate time/distance limit
  - Now defined more accurately

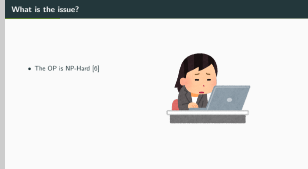
What is the issue?

What is the issue?

- So what are we here to talk about?

What is the issue?

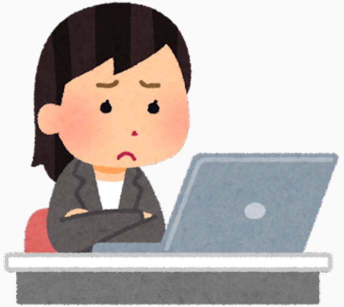
- The OP is NP-Hard [6]

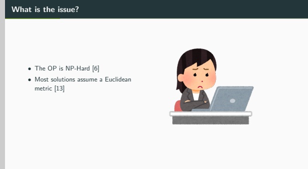


- Unsurprisingly, OP is NP-Hard
  - Okay, so there are are some approximate solutions, right?

What is the issue?

- The OP is NP-Hard [6]
- Most solutions assume a Euclidean metric [13]

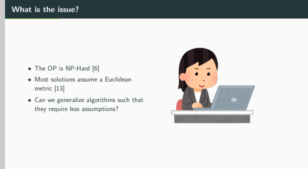




- Yes, but many solutions in the literature require Euclidean metric
  - Nodes are points in euclidean plane
  - Distances are euclidean distance
  - Most assume at least some amount of restrictions
- Not good if trying to solve general instances

What is the issue?

- The OP is NP-Hard [6]
- Most solutions assume a Euclidean metric [13]
- Can we generalize algorithms such that they require less assumptions?



- Can we generalize algorithms so the need less restrictions?

The Orienteering Problem

The Orienteering Problem

Definition (Orienteering Problem [13])
Let $G = (V = \{v_1, \ldots, v_n\}, E)$ be an undirected graph with a cost function $t : E \to \mathbb{R}_+$ and a profit function $s : V \to \mathbb{R}$. Also, let $T_{max} \in \mathbb{R}_+$ be a cost limit.

The Orienteering Problem

Definition (Orienteering Problem [13])
Let $G = (V = \{v_1, \ldots, v_n\}, E)$ be an undirected graph with a cost function $t : E \to \mathbb{R}_+$ and a profit function $s : V \to \mathbb{R}$. Also, let $T_{max} \in \mathbb{R}_+$ be a cost limit.

- Proper definition of the OP
  - Start with an undirected graph
  - Cost function t for the edges
    - Also “distance”, “travel time”, “weight”
  - Profit function s for the nodes
    - Also “score”
  - Maximum cost T_{max}

The Orienteering Problem

Definition (Orienteering Problem [13])
Let $G = (V = \{v_1, \dots, v_n\}, E)$ be an undirected graph with a cost function $t : E \to \mathbb{R}_+$ and a profit function $s : V \to \mathbb{R}$. Also, let $T_{max} \in \mathbb{R}_+$ be a cost limit.

The $\textsc{OrienteeringProblem}$ aims at finding a path $P = [p_1, \dots, p_k], p_i \in V$ with $p_1 = v_1$ and ending at $p_k = v_n$ which maximizes the total profit $S(P) := \sum_{p_i \in P} s(p_i)$ while respecting the cost limit, that is

$$T(P) := \sum_{i=1}^{k-1} t(v_i, v_{i+1}) \leq T_{max}$$

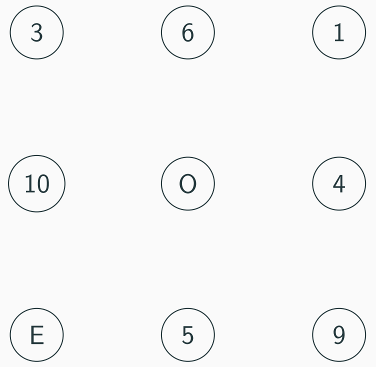
The Orienteering Problem

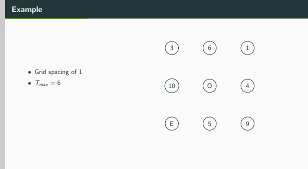
Definition (Orienteering Problem [13])
Let $G = (V = \{v_1, \dots, v_n\}, E)$ be an undirected graph with a cost function $t : E \to \mathbb{R}_+$ and a profit function $s : V \to \mathbb{R}$. Also, let $T_{max} \in \mathbb{R}_+$ be a cost limit.
The $\textsc{OrienteeringProblem}$ aims at finding a path $P = [p_1, \dots, p_k], p_i \in V$ with $p_1 = v_1$ and ending at $p_k = v_n$ which maximizes the total profit $S(P) := \sum_{p_i \in P} s(p_i)$ while respecting the cost limit, that is
$$T(P) := \sum_{i=1}^{k-1} t(v_i, v_{i+1}) \leq T_{max}$$

- Orienteering Problem
  - Want a path from v_1 to v_n
    - Need not be fixed but often are
    - Named like this for convenience
  - Path should maximize the total profit...
  - ...while respecting the cost limit T_{max}
- Questions?

Example

- Grid spacing of 1
- $T_{max} = 6$

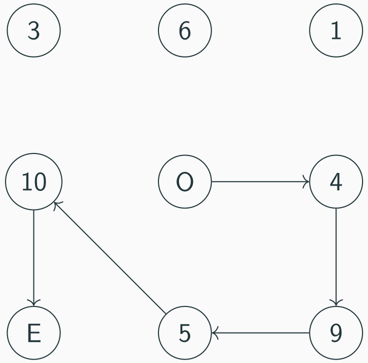


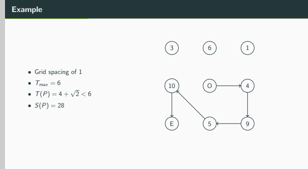


- Example OP instance
  - $T_{max} = 6$
  - Edges between all nodes (omitted for clarity)
  - Horizontal/Vertical distance between neighboring nodes: 1
  - Diagonal movement: $\sqrt{2} \approx 1.4$
- Ask audience: What is the optimal path?

Example

- Grid spacing of 1
- $T_{max} = 6$
- $T(P) = 4 + \sqrt{2} < 6$
- $S(P) = 28$





- Best path as far as I can see

Complete Graphs

Complete Graphs

- Getting into the problem
  - Restrictions on allowed input graphs
  - Introduce common ones in order of strictness
  - After: introduce algorithms and try to reduce the restrictions they require

Complete Graphs

Definition (Complete Graphs)
A graph $G = (V, E)$ is *complete* if for any two nodes $v_i \neq v_j \in V$ there exists a corresponding edge $\{v_i, v_j\} \in E$.

Complete Graphs
Definition (Complete Graphs)
A graph $G = (V, E)$ is complete if for any two nodes $v_i \neq v_j \in V$ there exists a corresponding edge $\{v_i, v_j\} \in E$.

- First: Complete Graph
- Probably known to everyone
  - But for completeness' sake

Complete Graphs

Definition (Complete Graphs)
A graph $G = (V, E)$ is *complete* if for any two nodes $v_i \neq v_j \in V$ there exists a corresponding edge $\{v_i, v_j\} \in E$.

- Almost unanimous in the literature [13, 7, 9, 11]

Complete Graphs
Definition (Complete Graphs)
A graph $G = (V, E)$ is complete if for any two nodes $v_i \neq v_j \in V$ there exists a corresponding edge $\{v_i, v_j\} \in E$.
- Almost unanimous in the literature [13, 7, 9, 11]

- First: Complete Graph
- Probably known to everyone
  - But for completeness' sake
- Almost unanimous

Complete Graphs

Definition (Complete Graphs)
A graph $G = (V, E)$ is *complete* if for any two nodes $v_i \neq v_j \in V$ there exists a corresponding edge $\{v_i, v_j\} \in E$.

- Almost unanimous in the literature [13, 7, 9, 11]
- Simple to transform any graph into a complete graph
  - Might be computationally expensive on large inputs

Complete Graphs
Definition (Complete Graphs)
A graph $G = (V, E)$ is complete if for any two nodes $v_i \neq v_j \in V$ there exists a corresponding edge $\{v_i, v_j\} \in E$.
- Almost unanimous in the literature [13, 7, 9, 11]
- Simple to transform any graph into a complete graph
  - Might be computationally expensive on large inputs

- First: Complete Graph
- Probably known to everyone
  - But for completeness' sake
- Almost unanimous
- Simple to transform any graph into complete graph. Insert missing edges with weight
  - Weight of the shortest path between nodes
  - ∞ if the missing edges should not be taken
    - Might not work for all algorithms though (example later)

Triangle Inequality

- First actual restriction
  - Similarly defined as in geometry

Triangle Inequality

Triangle Inequality

Definition (Triangle Inequality [1])
A complete, weighted graph $G = (V, E)$ satisfies the *triangle inequation* if any three nodes $u \neq v \neq w \in V$ statisfy the following inequality:

$$t(u, w) \leq t(u, v) + t(u, v)$$

Triangle Inequality
Definition (Triangle Inequality [1])
A complete, weighted graph $G = (V, E)$ satisfies the *triangle inequation* if any three nodes $u \neq v \neq w \in V$ statisfy the following inequality:
$t(u, w) \leq t(u, v) + t(u, v)$

- First actual restriction
  - Similarly defined as in geometry
- Any three nodes' weights satisfy the triangle inequality

Triangle Inequality

Definition (Triangle Inequality [1])
A complete, weighted graph $G = (V, E)$ satisfies the *triangle inequation* if any three nodes $u \neq v \neq w \in V$ statisfy the following inequality:

$$t(u, w) \leq t(u, v) + t(u, v)$$

$\Rightarrow$ The direct edge is always the shortest path between nodes.

Triangle Inequality
Definition (Triangle Inequality [1])
A complete, weighted graph $G = (V, E)$ satisfies the *triangle inequation* if any three nodes $u \neq v \neq w \in V$ statisfy the following inequality:
$t(u, w) \leq t(u, v) + t(u, v)$
$\Rightarrow$ The direct edge is always the shortest path between nodes.

- First actual restriction
  - Similarly defined as in geometry
- Any three nodes' weights satisfy the triangle inequality
- $\Rightarrow$ The direct edge is always the shortest path between nodes.
  - No detour will ever be faster

Triangle Inequality

Definition (Triangle Inequality [1])
A complete, weighted graph $G = (V, E)$ satisfies the *triangle inequation* if any three nodes $u \neq v \neq w \in V$ statisfy the following inequality:

$$t(u, w) \leq t(u, v) + t(u, v)$$

$\Rightarrow$ The direct edge is always the shortest path between nodes.

- Less often explicitly stated [9]
- More often implied by requiring a Euclidean metric

Triangle Inequality
Definition (Triangle Inequality [1])
A complete, weighted graph $G = (V, E)$ satisfies the *triangle inequation* if any three nodes $u \neq v \neq w \in V$ statisfy the following inequality:
$t(u, w) \leq t(u, v) + t(u, v)$
$\Rightarrow$ The direct edge is always the shortest path between nodes.
- Less often explicitly stated [9]
- More often implied by requiring a Euclidean metric

- First actual restriction
  - Similarly defined as in geometry
- Any three nodes' weights satisfy the triangle inequality
- $\Rightarrow$ The direct edge is always the shortest path between nodes.
  - No detour will ever be faster
- Less often explicitly stated in literature
- Usually implied as part of Euclidean Metric
  - Defined shortly

Triangle Inequality

- What do we get out of this?

Triangle Inequality

Triangle Inequality

- Many problems have better approximations on such graphs [1]
- Easier to reason about the rest of the graph.

Triangle Inequality
- Many problems have better approximations on such graphs [1]
- Easier to reason about the rest of the graph.

- Many problems: better approximations on more restricted graphs
- Easier to reason about rest of graph
  - example following now

Triangle Inequality

- Many problems have better approximations on such graphs [1]
- Easier to reason about the rest of the graph.
- Simple greedy algorithm

- Nodes fulfilling the former property are called *available*

Triangle Inequality
- Many problems have better approximations on such graphs [1]
- Easier to reason about the rest of the graph.
- Simple greedy algorithm
- Nodes fulfilling the former property are called available

- Many problems: better approximations on more restricted graphs
- Easier to reason about rest of graph
  - example following now
- Simple greedy algorithm

Triangle Inequality

- Many problems have better approximations on such graphs [1]
- Easier to reason about the rest of the graph.
- Simple greedy algorithm
  - Let $P = [p_1, \dots, p_k]$ is the current path
  - Always pick the node $v \in V \setminus P$ such that

$$T(P \cup [v, v_n]) \leq T_{max}$$

  and

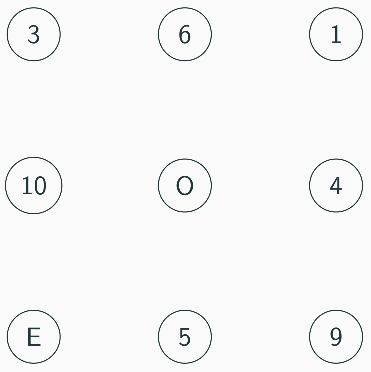
$$\frac{s(v)}{t(p_k, v)}$$

  is maximized.
- Nodes fulfilling the former property are called *available*
- If no such node exists, go to the end node.

Triangle Inequality
Many problems have better approximations on such graphs [1]
Easier to reason about the rest of the graph.
Simple greedy algorithm
Let $P = [p_1, \dots, p_k]$ is the current path
Always pick the node $v \in V \setminus P$ such that
$T(P \cup [v, v_n]) \leq T_{max}$
and $\frac{s(v)}{t(p_k, v)}$
is maximized.
Nodes fulfilling the former property are called available
If no such node exists, go to the end node.

- Many problems: better approximations on more restricted graphs
- Easier to reason about rest of graph
  - example following now
- Simple greedy algorithm
- We always want to pick a node per step
  - Navigating to that node and then to the end may not violate T_{max}
  - Maximize the ratio between score gained and distance traveled
- If no such node, go to the end

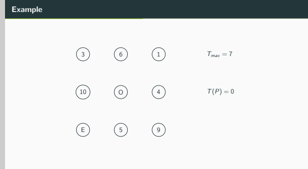
Example



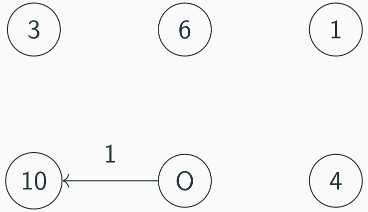
$T_{max} = 7$

$T(P) = 0$

2023-01-20

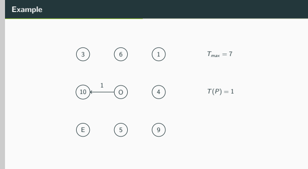


Example

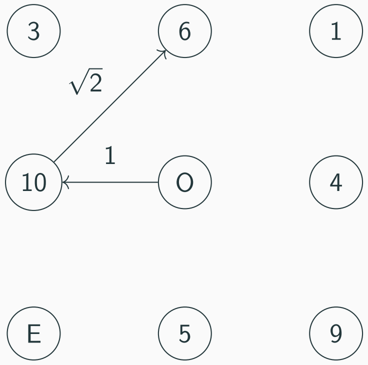


$T_{max} = 7$

$T(P) = 1$

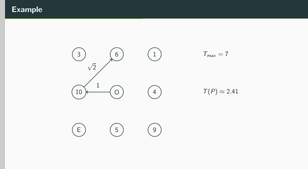


Example

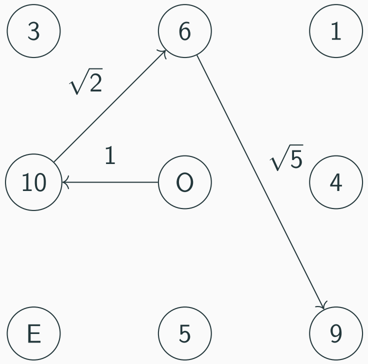


$T_{max} = 7$

$T(P) \approx 2.41$

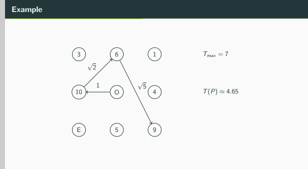


Example



$T_{max} = 7$

$T(P) \approx 4.65$



Z.B. jetzt

- Remaining capacity of around 2.35.
- To which node can we go (and then to the end) without volating T_{max}?

$\Rightarrow$ Only the node 5

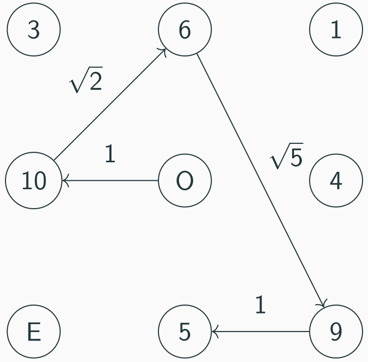
Technically we could travel to node 4 instead without immediately violating T_{max}

- But travelling to the end then would violate it.
- Since we know this: Triangle Inequality $\Rightarrow$ There is no shorter way

$\Rightarrow$ Travelling to 4 does not allow reaching the end

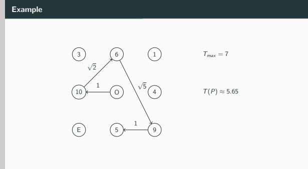
Both algorithms presented today, rely on this fact

Example



$T_{max} = 7$

$T(P) \approx 5.65$



Z.B. jetzt

- Remaining capacity of around 2.35.
- To which node can we go (and then to the end) without volating T_{max}?

$\Rightarrow$ Only the node 5

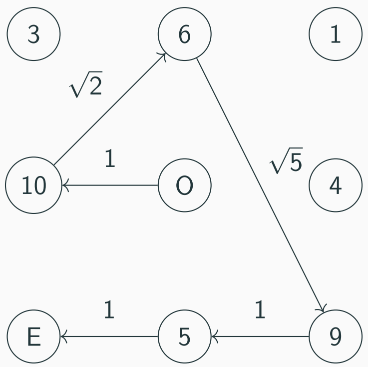
Technically we could travel to node 4 instead without immediately violating T_{max}

- But travelling to the end then would violate it.
- Since we know this: Triangle Inequality $\Rightarrow$ There is no shorter way

$\Rightarrow$ Travelling to 4 does not allow reaching the end

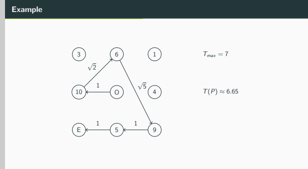
Both algorithms presented today, rely on this fact

Example



$T_{max} = 7$

$T(P) \approx 6.65$



Z.B. jetzt

- Remaining capacity of around 2.35.
- To which node can we go (and then to the end) without volating T_{max}?

$\Rightarrow$ Only the node 5

Technically we could travel to node 4 instead without immediately violating T_{max}

- But travelling to the end then would violate it.
- Since we know this: Triangle Inequality $\Rightarrow$ There is no shorter way

$\Rightarrow$ Travelling to 4 does not allow reaching the end

Both algorithms presented today, rely on this fact

Euclidean Metric

Definition
For every node $v_i \in V$, there are coordinates $(x_i, y_i)^\intercal \in \mathbb{R}^2$ and for each pair of nodes $v_i \neq v_j \in V$ the edge weights are defined as:

$$t(v_i, v_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Euclidean Metric
Definition
For every node $v_i \in V$, there are coordinates $(x_i, y_i)^\intercal \in \mathbb{R}^2$ and for each pair of nodes $v_i \neq v_j \in V$ the edge weights are defined as:
$t(v_i, v_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

Definition like in geometry Nodes are points in the euclidean plane

- Edge weights are distances between them

Euclidean Metric

Definition
For every node $v_i \in V$, there are coordinates $(x_i, y_i)^\intercal \in \mathbb{R}^2$ and for each pair of nodes $v_i \neq v_j \in V$ the edge weights are defined as:

$$t(v_i, v_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Entails the previous two restrictions
- Very common in the literature [5, 6, 11, 12]

Euclidean Metric
Definition
For every node $v_i \in V$, there are coordinates $(x_i, y_i)^\intercal \in \mathbb{R}^2$ and for each pair of nodes $v_i \neq v_j \in V$ the edge weights are defined as:
$t(v_i, v_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
- Entails the previous two restrictions
- Very common in the literature [5, 6, 11, 12]

Definition like in geometry Nodes are points in the euclidean plane

- Edge weights are distances between them

Note, that this entails completeness and the triangle inequality

$\Rightarrow$ Entails the previous two restrictions

Commonly seen in the literature

Tsiligiridis' Algorithms

Tsiligiridis' Algorithms

Tsiligiridis' Algorithms

- One of the first papers on the topic [12]

Tsiligiridis' Algorithms

- One of the first papers on the topic [12]

One of the first papers on the OP

Tsiligiridis' Algorithms

- One of the first papers on the topic [12]
- Introduced two path-generating algorithms and one path-improving algorithm

Tsiligiridis' Algorithms
- One of the first papers on the topic [12]
- Introduced two path-generating algorithms and one path-improving algorithm

One of the first papers on the OP Introduced two path-generating algorithms

- Stochastic algorithm
- Deterministic algorithm

Introduced one path-improving algorithm

- Route Improving algorithm

Tsiligiridis' Algorithms

- One of the first papers on the topic [12]
- Introduced two path-generating algorithms and one path-improving algorithm
- In the original paper the input is assumed to be Euclidean

Tsiligiridis' Algorithms
- One of the first papers on the topic [12]
- Introduced two path-generating algorithms and one path-improving algorithm
- In the original paper the input is assumed to be Euclidean

One of the first papers on the OP Introduced two path-generating algorithms

- Stochastic algorithm
- Deterministic algorithm

Introduced one path-improving algorithm

- Route Improving algorithm

In original paper: Input assumed to be euclidean

- test instances used are also euclidean

Tsiligiridis' Algorithms

- One of the first papers on the topic [12]
- Introduced two path-generating algorithms and one path-improving algorithm
- In the original paper the input is assumed to be Euclidean
- Will take a look at one algorithm of each category

Tsiligiridis' Algorithms

- One of the first papers on the topic [12]
- Introduced two path-generating algorithms and one path-improving algorithm
- In the original paper the input is assumed to be Euclidean
- Will take a look at one algorithm of each category

One of the first papers on the OP Introduced two path-generating algorithms

- Stochastic algorithm
- Deterministic algorithm

Introduced one path-improving algorithm

- Route Improving algorithm

In original paper: Input assumed to be euclidean

- test instances used are also euclidean

Will discuss the S algorithm and RI algorithm

Stochastic Algorithm (S-Algorithm)

- Similar to the simple example algorithm we looked at before

Stochastic Algorithm (S-Algorithm)
- Similar to the simple example algorithm we looked at before

Stochastic Algorithm (S-Algorithm)

- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how “desirable” it is

Stochastic Algorithm (S-Algorithm)
• Similar to the simple example algorithm we looked at before
• Always pick a random node based on how “desirable” it is

Instead of picking the locally “best” node

- Weigh each available node by its desirability and pick one randomly

Stochastic Algorithm (S-Algorithm)

- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how “desirable” it is

Definition (Desirability)
Let $P := [p_1, \dots, p_l]$ be an unfinished path and $r \in \mathbb{R}_+$ a constant. If $A \subseteq V \setminus P$ is the set of available nodes, then the *desirability value* $D(v_i)$ for every $v_i \in A$ is

$$D(v_i) := \left(\frac{s(v_i)}{t(p_l, v_i)}\right)^r$$

Stochastic Algorithm (S-Algorithm)
- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how “desirable” it is
Definition (Desirability)
Let $P := [p_1, \dots, p_l]$ be an unfinished path and $r \in \mathbb{R}_+$ a constant. If $A \subseteq V \setminus P$ is the set of available nodes, then the desirability value $D(v_i)$ for every $v_i \in A$ is
$D(v_i) := \left(\frac{s(v_i)}{t(p_l, v_i)}\right)^r$

Instead of picking the locally “best” node

- Weigh each available node by its desirability and pick one randomly

We have some unfinished path and a constant r.

- Desirability defined almost the same as before.
- Measure for how valuable a node is estimated to be.

Stochastic Algorithm (S-Algorithm)

- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how “desirable” it is

Definition (Desirability)
Let $P := [p_1, \ldots, p_l]$ be an unfinished path and $r \in \mathbb{R}_+$ a constant. If $A \subseteq V \setminus P$ is the set of available nodes, then the *desirability value* $D(v_i)$ for every $v_i \in A$ is

$$D(v_i) := \left(\frac{s(v_i)}{t(p_l, v_i)}\right)^r$$

- In each step consider the k most desirable nodes for some $k \in \mathbb{N}$

Stochastic Algorithm (S-Algorithm)
- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how “desirable” it is
Definition (Desirability)
Let $P := [p_1, \ldots, p_l]$ be an unfinished path and $r \in \mathbb{R}_+$ a constant. If $A \subseteq V \setminus P$ is the set of available nodes, then the desirability value $D(v_i)$ for every $v_i \in A$ is
$$D(v_i) := \left(\frac{s(v_i)}{t(p_l, v_i)}\right)^r$$
- In each step consider the k most desirable nodes for some $k \in \mathbb{N}$

Instead of picking the locally “best” node

- Weigh each available node by its desirability and pick one randomly

We have some unfinished path and a constant r.

- Desirability defined almost the same as before.
- Measure for how valuable a node is estimated to be.

We need another parameter k

- We always only consider the best k nodes when choosing a next node

Stochastic Algorithm (S-Algorithm)

- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how "desirable" it is

Definition (Desirability)
Let $P := [p_1, \ldots, p_l]$ be an unfinished path and $r \in \mathbb{R}_+$ a constant. If $A \subseteq V \setminus P$ is the set of available nodes, then the *desirability value* $D(v_i)$ for every $v_i \in A$ is

$$D(v_i) := \left(\frac{s(v_i)}{t(p_l, v_i)}\right)^r$$

- In each step consider the k most desirable nodes for some $k \in \mathbb{N}$
- If $A_k \subseteq A$ is the set of the k most desirable nodes then the probability for a node $v_i \in A_k$ is:

$$\frac{D(v_i)}{\sum_{v \in A_k} D(v)}$$

Stochastic Algorithm (S-Algorithm)
- Similar to the simple example algorithm we looked at before
- Always pick a random node based on how "desirable" it is
Definition (Desirability)
Let $P := [p_1, \ldots, p_l]$ be an unfinished path and $r \in \mathbb{R}_+$ a constant. If $A \subseteq V \setminus P$ is the set of available nodes, then the desirability value $D(v_i)$ for every $v_i \in A$ is
$$D(v_i) := \left(\frac{s(v_i)}{t(p_l, v_i)}\right)^r$$
- In each step consider the k most desirable nodes for some $k \in \mathbb{N}$
- If $A_k \subseteq A$ is the set of the k most desirable nodes then the probability for a node $v_i \in A_k$ is:
$$\frac{D(v_i)}{\sum_{v \in A_k} D(v)}$$

Instead of picking the locally "best" node

- Weigh each available node by its desirability and pick one randomly

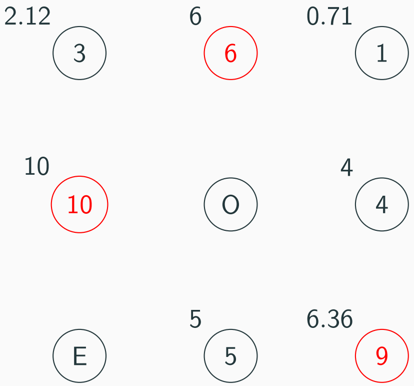
We have some unfinished path and a constant r.

- Desirability defined almost the same as before.
- Measure for how valuable a node is estimated to be.

We need another parameter k

- We always only consider the best k nodes when choosing a next node

Example

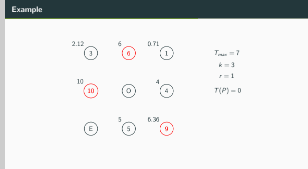


$T_{max} = 7$

$k = 3$

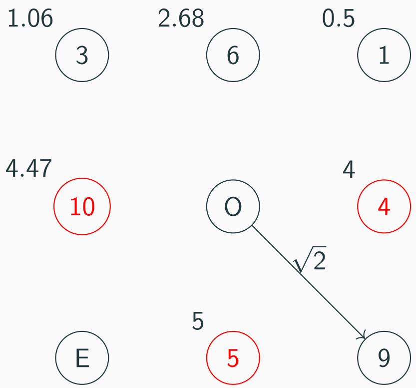
$r = 1$

$T(P) = 0$



- Start at the origin
  - Nodes on a grid with horiz/vert. distance of 1
  - $\Rightarrow$ Diagonal $= \sqrt{2} \approx 1.41$, long diagonal $= \sqrt{5} \approx 2.24$
  - Always consider the 3 best nodes
  - We set $r = 1$ to simplify calculations
  - Score inside nodes, Desirability on the sides
- First step: $10, 6$ and 9 are the best.
  - We randomly pick 9

Example

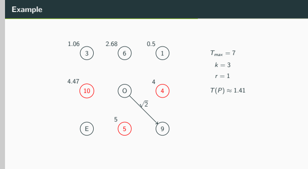


$T_{max} = 7$

$k = 3$

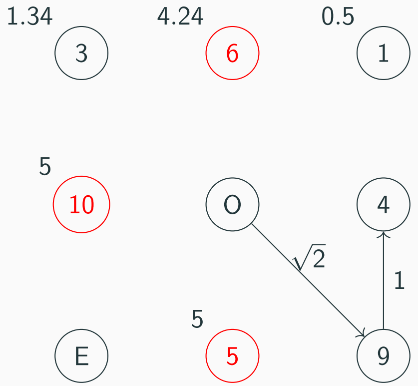
$r = 1$

$T(P) \approx 1.41$



- Now we continue like this

Example

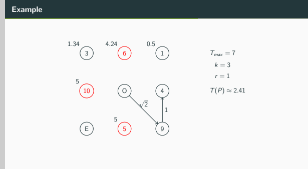


$T_{max} = 7$

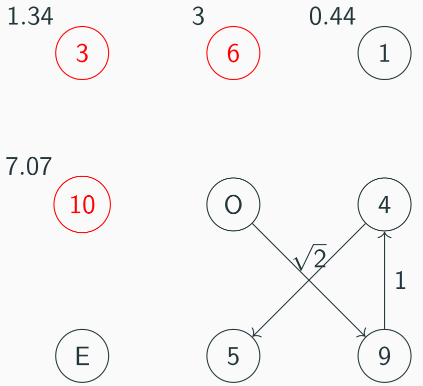
$k = 3$

$r = 1$

$T(P) \approx 2.41$



Example

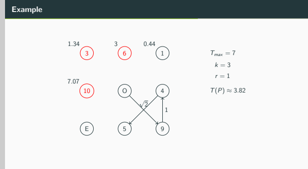


$T_{max} = 7$

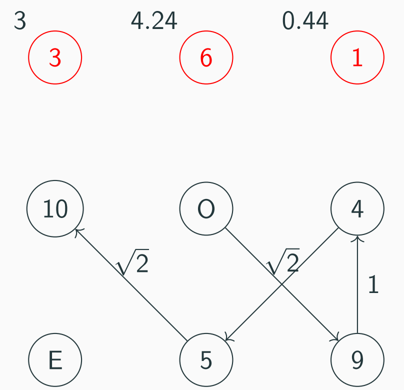
$k = 3$

$r = 1$

$T(P) \approx 3.82$



Example

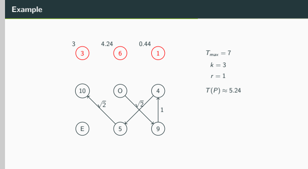


$T_{max} = 7$

$k = 3$

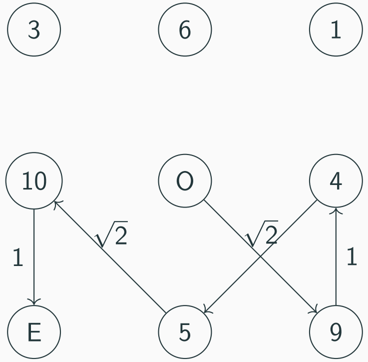
$r = 1$

$T(P) \approx 5.24$



- Note in this step:
  - 3 nodes still unvisited
  - But node 3 would require 1 to go there and 2 to the end
  - Other two nodes even further away
  - No choice but to go to the end node
- Note that this algorithm decides this based on the triangle equality like the example algorithm before.

Example

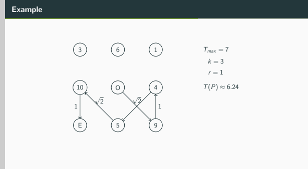


$T_{max} = 7$

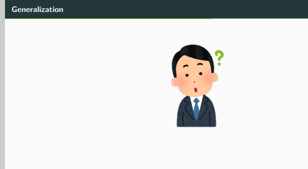
$k = 3$

$r = 1$

$T(P) \approx 6.24$



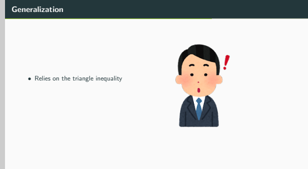
Generalization



- Main point: Trying to drop as many restrictions as feasibly possible
  - So which restrictions does the algorithm rely on

Generalization

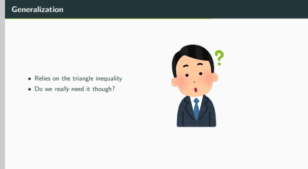
- Relies on the triangle inequality



- Main point: Trying to drop as many restrictions as feasibly possible
  - So which restrictions does the algorithm rely on
- Nowhere are uniquely Euclidean features ever required
  - Selecting nodes relies on triangle equality though

Generalization

- Relies on the triangle inequality
- Do we *really* need it though?

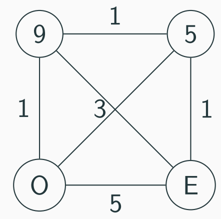


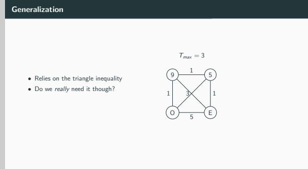
- Main point: Trying to drop as many restrictions as feasibly possible
  - So which restrictions does the algorithm rely on
- Nowhere are uniquely Euclidean features ever required
  - Selecting nodes relies on triangle equality though
- Do we really need it?
  - What happens if we drop the triangle inequality

Generalization

$T_{max} = 3$

- Relies on the triangle inequality
- Do we *really* need it though?

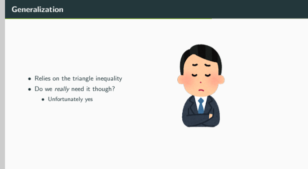




- Starting at the origin, which nodes can we go to? (Algorithm's perspective)
  - 9? No, since going to 9 and then to the goal would cost 4
  - 5? No, since going to 5 and then to the goal would cost 4
  - The goal? No, it would cost 5
- Would return, that there is no path
  - Obviously wrong

Generalization

- Relies on the triangle inequality
- Do we *really* need it though?
  - Unfortunately yes



- Seems like there is no trivial way around it
  - Calculate the shortest path to the end for every node?
    - Might work to an extent
    - ! But shortest paths assume all nodes to be usable
    - If some nodes of the paths are already used, the shortest path lengths might increase

Szwarc-Boryczka Algorithm

Szwarc-Boryczka Algorithm

- Let's go to a more recent algorithm
- Slight detour to introduce the algorithm's foundaition

Harmony Search [3, 4, 5]

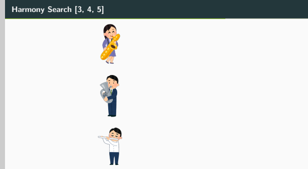
Harmony Search [3, 4, 5]

- Harmony Search
  - Optimization Technique proposed by Geem in 2000
  - Name stems from analogy it is inspired by
    - Maybe a little weird at first but makes more sense by the end (imo)

Harmony Search [3, 4, 5]

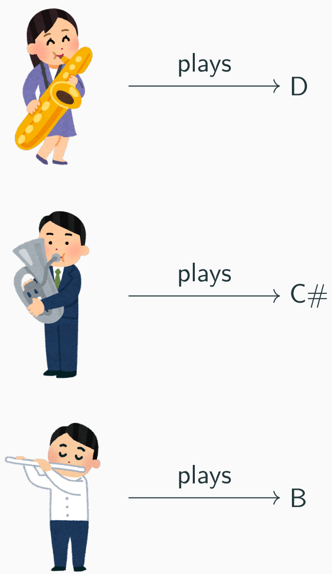
- Imagine band during an improvisation session

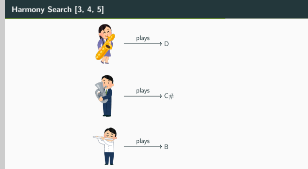
Harmony Search [3, 4, 5]



- Several individual players playing notes (or pitches as called here)
  - Resulting “collection” of pitches results in a *harmony*

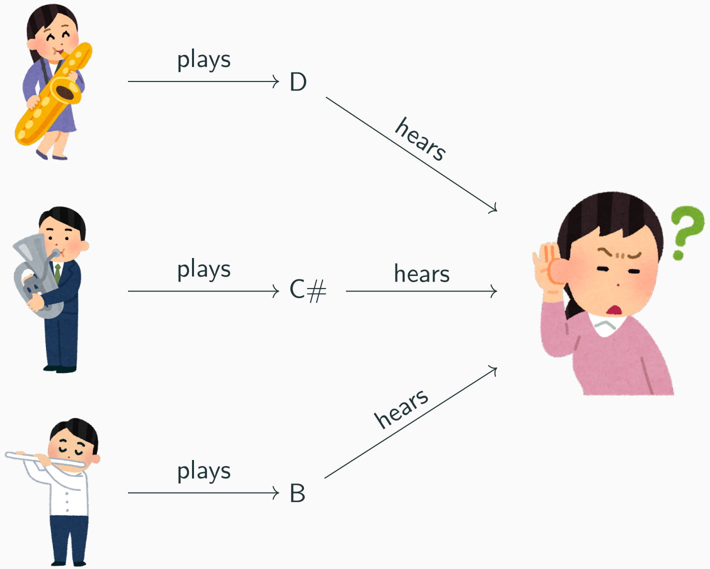
Harmony Search [3, 4, 5]

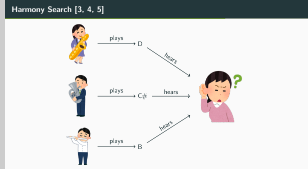




- Several individual players playing notes (or pitches as called here)
  - Resulting "collection" of pitches results in a *harmony*
- At first kind of poorly coordinated
  - Resulting harmonies might be sort of random

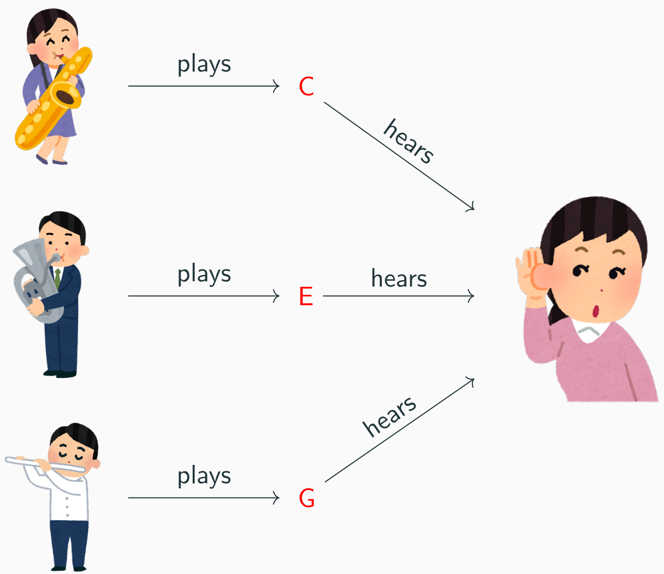
Harmony Search [3, 4, 5]

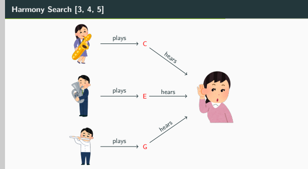




- Several individual players playing notes (or pitches as called here)
  - Resulting “collection” of pitches results in a *harmony*
- At first kind of poorly coordinated
  - Resulting harmonies might be sort of random

Harmony Search [3, 4, 5]





- Several individual players playing notes (or pitches as called here)
  - Resulting “collection” of pitches results in a *harmony*
- At first kind of poorly coordinated
  - Resulting harmonies might be sort of random

Literature i

[1] Paul E. Black. *Triangle Inequality*. In: *Dictionary of Algorithms and Data Structures*. CRC, Dec. 17, 2004. URL: `https://xlinux.nist.gov/dads/HTML/trianglnqlty.html` (visited on 11/18/2022).

[2] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem." In: *Management Science* 6.1 (Oct. 1959). Publisher: INFORMS, pp. 80–91. ISSN: 0025-1909. DOI: `10.1287/mnsc.6.1.80`. URL: `https://pubsonline.informs.org/doi/abs/10.1287/mnsc.6.1.80` (visited on 01/13/2023).

[3] Zong Woo Geem. "Optimal design of water distribution networks using harmony search." PhD thesis. Seoul: Korea University, 2000.

Literature i

[1] Paul E. Black. *Triangle Inequality*. In: *Dictionary of Algorithms and Data Structures*. CRC, Dec. 17, 2004. URL: https://xlinux.nist.gov/dads/HTML/trianglnqlty.html (visited on 11/18/2022).

[2] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem." In: *Management Science* 6.1 (Oct. 1959). Publisher: INFORMS, pp. 80–91. ISSN: 0025-1909. DOI: 10.1287/mnsc.6.1.80. URL: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.6.1.80 (visited on 01/13/2023).

[3] Zong Woo Geem. "Optimal design of water distribution networks using harmony search." PhD thesis. Seoul: Korea University, 2000.

Literature ii

[4] Zong Woo Geem. "State-of-the-Art in the Structure of Harmony Search Algorithm." In: *Recent Advances In Harmony Search Algorithm*. Ed. by Zong Woo Geem. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2010, pp. 1–10. ISBN: 978-3-642-04317-8. DOI: 10.1007/978-3-642-04317-8_1. URL: https://doi.org/10.1007/978-3-642-04317-8_1 (visited on 11/30/2022).

[5] Zong Woo Geem, Chung-Li Tseng, and Yongjin Park. "Harmony Search for Generalized Orienteering Problem: Best Touring in China." In: Lecture Notes in Computer Science. Vol. 3612. Aug. 27, 2005, pp. 439–439. ISBN: 978-3-540-28320-1. DOI: 10.1007/11539902_91.

Literature ii

[4] Zong Woo Geem. "State-of-the-Art in the Structure of Harmony Search Algorithm." In: *Recent Advances In Harmony Search Algorithm*. Ed. by Zong Woo Geem. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2010, pp. 1–10. ISBN: 978-3-642-04317-8. DOI: 10.1007/978-3-642-04317-8_1. URL: https://doi.org/10.1007/978-3-642-04317-8_1 (visited on 11/30/2022).

[5] Zong Woo Geem, Chung-Li Tseng, and Yongjin Park. "Harmony Search for Generalized Orienteering Problem: Best Touring in China." In: Lecture Notes in Computer Science. Vol. 3612. Aug. 27, 2005, pp. 439–439. ISBN: 978-3-540-28320-1. DOI: 10.1007/11539902_91.

[6] Bruce L. Golden, Larry Levy, and Rakesh Vohra. "The orienteering problem." In: *Naval Research Logistics (NRL)* 34.3 (1987), pp. 307–318. ISSN: 1520-6750. DOI: 10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28198706%2934%3A3%3C307%3A%3AAID-NAV3220340302%3E3.0.CO%3B2-D (visited on 11/10/2022).

[7] Gilbert Laporte and Silvano Martello. "The selective travelling salesman problem." In: *Discrete Applied Mathematics* 26.2 (Mar. 1, 1990), pp. 193–207. ISSN: 0166-218X. DOI: 10.1016/0166-218X(90)90100-Q. URL: https://www.sciencedirect.com/science/article/pii/0166218X9090100Q (visited on 11/10/2022).

Literature iii

[6] Bruce L. Golden, Larry Levy, and Rakesh Vohra. "The orienteering problem." In: *Naval Research Logistics (NRL)* 34.3 (1987), pp. 307–318. ISSN: 1520-6750. DOI: 10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28198706%2934%3A3%3C307%3A%3AAID-NAV3220340302%3E3.0.CO%3B2-D (visited on 11/10/2022).

[7] Gilbert Laporte and Silvano Martello. "The selective travelling salesman problem." In: *Discrete Applied Mathematics* 26.2 (Mar. 1, 1990), pp. 193–207. ISSN: 0166-218X. DOI: 10.1016/0166-218X(90)90100-Q. URL: https://www.sciencedirect.com/science/article/pii/0166218X9090100Q (visited on 11/10/2022).

[8] Martin Placek. *Amazon Logistics: Package volume in the U.S.* Publication Title: Statista. Nov. 2022. URL: https://www.statista.com/statistics/1178979/amazon-logistics-package-volume-united-states/.

[9] Alberto Santini and Claudia Archetti. "The hazardous orienteering problem." In: *Networks* n/a (n/a Nov. 8, 2022). ISSN: 1097-0037. DOI: 10.1002/net.22129. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22129 (visited on 11/19/2022).

[10] Alexander Schrijver. "On the history of combinatorial optimization (till 1960)." In: *Handbook of Discrete Optimization*. Vol. 12. Handbooks in Operations Research and Management Science. Amsterdam: Elsevier, Jan. 2005, pp. 1–68. ISBN: 978-0-444-51507-0. URL: http://homepages.cwi.nl/~lex/files/histco.pdf (visited on 01/13/2023).

Literature iv

[8] Martin Placek. *Amazon Logistics: Package volume in the U.S.* Publication Title: Statista. Nov. 2022. URL: https://www.statista.com/statistics/1178979/amazon-logistics-package-volume-united-states/.

[9] Alberto Santini and Claudia Archetti. "The hazardous orienteering problem." In: *Networks* n/a (n/a Nov. 8, 2022). ISSN: 1097-0037. DOI: 10.1002/net.22129. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22129 (visited on 11/19/2022).

[10] Alexander Schrijver. "On the history of combinatorial optimization (till 1960)." In: *Handbook of Discrete Optimization*. Vol. 12. Handbooks in Operations Research and Management Science. Amsterdam: Elsevier, Jan. 2005, pp. 1–68. ISBN: 978-0-444-51507-0. URL: http://homepages.cwi.nl/~lex/files/histco.pdf (visited on 01/13/2023).

[11] Krzysztof Szwarc and Urszula Boryczka. “A novel approach to the Orienteering Problem based on the Harmony Search algorithm.” In: *PLOS ONE* 17.2 (Feb. 28, 2022). Publisher: Public Library of Science, e0264584. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0264584. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0264584 (visited on 11/04/2022).

[12] Theodore Tsiligiridis. “Heuristic Methods Applied to Orienteering.” In: *Journal of the Operational Research Society* 35 (Sept. 1, 1984), pp. 797–809. DOI: 10.1057/jors.1984.162.

Literature v

[11] Krzysztof Szwarc and Urszula Boryczka. “A novel approach to the Orienteering Problem based on the Harmony Search algorithm.” In: *PLOS ONE* 17.2 (Feb. 28, 2022). Publisher: Public Library of Science, e0264584. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0264584. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0264584 (visited on 11/04/2022).

[12] Theodore Tsiligiridis. “Heuristic Methods Applied to Orienteering.” In: *Journal of the Operational Research Society* 35 (Sept. 1, 1984), pp. 797–809. DOI: 10.1057/jors.1984.162.

Literature vi

[13] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. "The orienteering problem: A survey." In: *European Journal of Operational Research* 209.1 (Feb. 16, 2011), pp. 1–10. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2010.03.045. URL: https://www.sciencedirect.com/science/article/pii/S0377221710002973 (visited on 11/04/2022).

Literature vi

[13] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. "The orienteering problem: A survey." In: *European Journal of Operational Research* 209.1 (Feb. 16, 2011), pp. 1–10. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2010.03.045. URL: https://www.sciencedirect.com/science/article/pii/S0377221710002973 (visited on 11/04/2022).