

technische universiteit eindhoven
Department of Mathematics and Computer Science

Master's Thesis

**Range Assignment
with Directional Antennas**

by
Fred van Nijnatten

Supervisor

prof. dr. M.T. de Berg

dr. A. Wolff

Referee

prof. dr. ing. G. Woeginger

Eindhoven, October 15, 2008

Contents

1	Introduction	5
2	Previous work	9
2.1	Problems on the line.	9
2.2	Strong connectivity	9
2.3	Connectivity	10
2.4	Broadcast	11
2.5	Bounded hop	11
2.6	Heterogeneous networks	12
2.7	Directional Antennas	12
3	Approach	13
4	The T^3-algorithm	15
4.1	Relation with relaxed TSP	15
4.2	Improved analysis of the T^3 -algorithm for points in the plane and $\alpha = 2$	19
4.3	Further improvement for points in the plane and $\alpha = 2$	24
4.4	Generalization to $\alpha \geq 2$	28
4.5	Lower bounds	30
5	Cycle-Insertion algorithm	35
5.1	NISE and NICE: two constant-factor approximations for TSP^α	37
5.2	NISE with double update	42
5.3	Lower bounds	45
6	NP-hardness	47

7 Experiments	53
7.1 Random tests	53
7.2 TSPLIB	54
8 Conclusion	57
References	62
Appendix	63
A.1 Additional proofs	63
A.2 Lower bound for the NISE-method	64
A.3 Lower bound for the NICE-method	67
A.4 Lower bound for the NI-algorithm	71
A.5 Results of the experiments on TSPLIB instances	73

Chapter 1

Introduction

A wireless network consists of nodes that can communicate with each other by means of radio signals. Unlike wired networks the topology of the network is not fixed in advance: by varying the transmission power a node can reach more or less nodes. This comes at a price however, since the required power for a node s to reach a node t scales with $|st|^\alpha$, where $|st|$ denotes the Euclidean distance between the nodes s and t and $\alpha \geq 1$ is the distance-power gradient. In an ideal situation $\alpha = 2$, but it can be more due to various environmental conditions such as buildings or mountains. If $\alpha > 1$ it can be more power-efficient to relay a message through a sequence of intermediate stations than to send the message directly. Sending directly might also not be possible if stations are too far apart, so typically the communication will take place through multi-hop transmission. Then it becomes desirable to find an assignment of transmission power to each node such that the total power consumption of the network is minimized and the resulting network satisfies properties that guarantee good communication.

Given a set of stations S , assign a range $r(s) \geq 0$ to each station $s \in S$. The cost of the range assignment is the overall power consumption

$$\text{cost}(r) = \sum_{s \in S} r(s)^\alpha.$$

A station s can send a message to a station t if $r(s) \geq |st|$. This leads to a directed communication graph $\vec{G}_r = (S, \vec{E}_r)$ where $\vec{E}_r = \{(s, t) \mid r(s) \geq |st|\}$ is the set of arcs induced by the range assignment r . Figure 1.1 shows an example.

For each range assignment problem $\text{cost}(r)$ is to be minimized under the condition that the resulting graph satisfies a property Π :

- Strongly Connected (SC): \vec{G}_r is strongly connected.
- Broadcast (B): Given a source station s , s can reach every other station in \vec{G}_r .

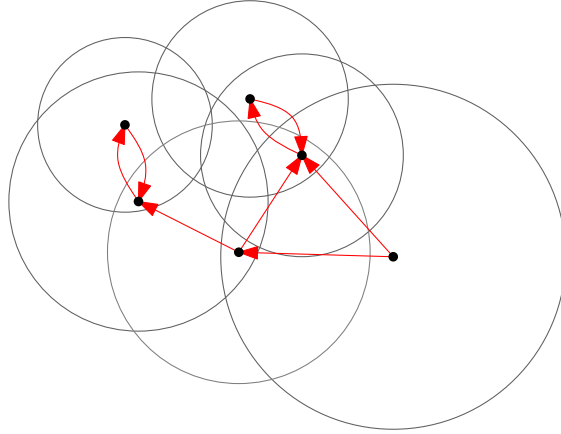


Figure 1.1: An example of a directed communication graph induced by a range assignment

Sometimes we would also like to bound the delay between sending and receiving. A communication suffers the most delay from processing in the nodes, while the delay for transmission is negligible. This gives rise to bounded-hops variants of the problems: for a given integer $h > 0$ every station must be able to reach any other station in at most h hops (hSC) or the source s must be able to reach any station in at most h hops (hB).

In network protocols it's often necessary to send acknowledgements back on a per-link basis. In the strongly connected communication graph this is not always possible. To make this possible the subgraph G_r of all bidirectional links $E = \{(s, t) \mid r(s) \geq |st| \text{ and } r(t) \geq |st|\}$ must be connected (C). In general all of these problems are NP-hard and therefore solutions must be sought in approximation algorithms.

Directed Antennas Most range assignment problems deal with omnidirectional antennas, but there are major advantages in wireless communications for using directional antennas. By concentrating the radio signal in one direction the transmitter can have a larger range and so reach more stations or it can use less energy to reach the same stations. There is also less chance for interference and eaves-dropping.

To cover the whole area, stations in a wireless network can be equipped with multiple antennas or the antenna can be steered mechanically in the desired direction. The first option can be expensive since the price scales with the number of antennas, but the latter has big disadvantages too: steering is less reliable, slower (the antenna must be turned first) and costs more energy. This disadvantage can be circumvented by turning the antennas only once such that all nodes can still communicate with each other using multiple hops. We would also like to minimize the energy by finding a suitable range assignment to all nodes. The strongly connected range assignment problem with directional antennas (dSC) is now as follows: Find an assignment to each node of the wireless network of the direction of the antenna and a range such that the resulting communication graph is strongly connected

and the total energy is minimized.

The cost of sending a message over a distance d with an omnidirectional antenna is proportional to d^α where $\alpha > 1$ is the distance-power gradient. We will assume that sending with a directional antenna in the plane over a distance d reaches all stations inside a circular sector of radius d and angle ϕ . Directing the antenna is part of the problem, but the angle is fixed. Figure 1.2 shows an example of a communication graph induced by an assignment of range and direction to a set of antennas.

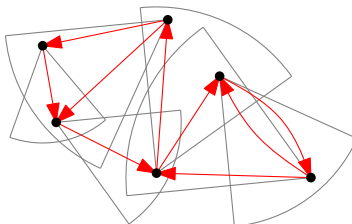


Figure 1.2: An example of a communication graph induced by an assignment of range and direction to directional antennas

The cost of sending is $\frac{\phi}{2\pi}d^\alpha$, which is only a fraction of the omnidirectional cost d^α . In general dimensions we'll assume that the cost is d^α/k for a constant $k \geq 1$. The total energy cost of a range assignment to a set of stations S is $1/k \sum_{s \in S} r^\alpha(s)$. The constant $1/k$ is irrelevant when optimizing this criterium, so from now on we'll use the same energy costs as for the omnidirectional antenna. The angle ϕ is still used to determine which stations can be reached by a station s .

Chapter 2

Previous work

There has already been a lot of research on range assignment problems. This chapter summarizes the main results and gives pointers to the literature.

2.1 Problems on the line.

When the stations are restricted to be on a line, most problems allow polynomial time solutions based on dynamic programming. The SC problem has been solved in $O(n^4)$ [29] and this has later been improved to $O(n^3)$ [22]. The hB problem has been solved in $O(hn^2)$ [15] and this has been improved to $O(n^2)$ [20]. For hSC only a 2-approximation is known [16]. For the approximation the related all-to-one problem on the line has been solved optimally in $O(hn^3)$. In the all-to-one problem all stations must be able to reach a *sink* node in at most h hops. The approximation then combines the solutions directed towards the leftmost and rightmost station by taking for each node the maximum of the ranges in the corresponding assignments.

In the rest of the paper we assume that the number of dimensions d is at least two.

2.2 Strong connectivity

The SC problem has been shown to be NP-hard for all values of α and APX-hard when $d \geq 3$ [25]. The problem remains NP-hard when restricted to well-spread instances. A family of instances is well-spread if $d_{\min} \leq c \cdot d_{\max}/\sqrt{n}$, where d_{\min}/d_{\max} is the minimum/maximum distance between any pair of points respectively and $c > 0$ is a constant. A simple 2-approximation exists based on computing a minimum spanning tree (MST) [29]. The algorithm computes an MST on the complete graph with edge weights set to $|st|^\alpha$. Then for each station the range is set to the maximum of the distances of the edges belonging to the station in the MST. We denote the corresponding range assignment with r_{MST} . So $\text{cost}(r_{\text{MST}}) = \sum_{v \in S} \max_{(v,u) \in \text{MST}} |uv|^\alpha$. The analysis of the approximation factor has been improved to $2 - \frac{2}{\lfloor \frac{n}{2} \rfloor + 1}$, which is tight for this algorithm [25].

For the case $\alpha = 1$ a $3/2$ -approximation exists [5]: compute an oriented minimum spanning tree towards a root node s and assign s the range that suffices to reach the farthest node from s . Repeat this construction for every node by varying the root node and take the minimum over all range assignments.

2.3 Connectivity

The MST algorithm already establishes bidirectional links and is also a 2-approximation for C [1]. The minimum-weight connected subgraph must be a spanning tree of the graph, so the problem is similar to finding a spanning tree T in the complete graph with minimal $\text{cost}(r_T)$ as defined above for MST. A way to lower the cost of a spanning tree is by using *forks*. A fork of a spanning tree consists of two edges with a common endpoint. The spanning tree can be improved by adding a fork and deleting the longest edges of the resulting cycles. The Greedy Fork Contraction (GFC) algorithm repeatedly selects the fork that lowers the cost of the current spanning tree most and then contracts the fork's nodes into a single node so that edges cannot be used again in the algorithm. The algorithm is a $11/6$ -approximation [1].

Greedy Prim-type and Kruskal-type heuristics have also been investigated: at each step add a node to the spanning tree component(s) that adds least to the current cost in an attempt to minimize $\text{cost}(r_T)$ directly. The Kruskal type heuristic does outperform MST in practice [1], but both heuristics have been shown to have the same worst-case behaviors when $\alpha > 1$, which are tight [25] also for SC. For $\alpha = 1$ it is not known whether the heuristics outperform MST. Interestingly, experiments have also shown that GFC is consistently outperformed by the Kruskal type heuristic when $n \leq 100$ [1]. The same authors also test a heuristic that performs edge and fork switching (EFS): at every step an edge or fork is added to the tree and the longest edges in the resulting cycle(s) are removed. Repeatedly the edge or fork with largest gain is selected until no further improvement is possible. In the test the optimum is computed for $n < 40$ by using an integer linear program formulation that solves an instance in 1 hour. The experiments show near-optimal behavior for EFS. In other experiments an heuristic that performs double edge switching (ES2) (add two, not necessary connected, edges to the tree and remove the longest edges from each cycle) is tested [36] and the authors claim a percentage improvement over MST that is twice as much as EFS. They also claim that ES2 still has a lower bound on the approximation factor of 2, but the proof is omitted due to page limit restrictions. This would be interesting because ES2 can be seen as a sort of generalization of GFC, but the latter has an upper bound on the approximation factor of $11/6$. A draft version of the upcoming journal version of the paper [37] contains an error in the proof.

2.4 Broadcast

The problem is trivial for $\alpha = 1$: simply assign a range to the source that is large enough to reach all stations in one hop and assign a range of 0 to all other stations. For $\alpha > 1$ the problem is NP-hard [25] and the most fundamental algorithm is again an MST-heuristic [42]: compute an MST as for SC, direct all nodes from the root towards the leafs and assign each node the range corresponding to the largest outgoing arc. The approximation ratio of MST is unbounded if $\alpha < d$ [14]. If $\alpha \geq d$ the approximation is constant for constant values of α . For $d = \alpha = 2$ the approximation ratio is 6 [2] and this is tight [41]. The ratio has been reduced progressively from 40 to 6 in several studies [14, 24, 30, 41, 33]. For $d = \alpha = 3$ the approximation factor is between 12 and 18.8 [34]. The same approximation factors also hold for $\alpha > d$ [24]. For $d > 3$ the approximation ratio is at most $3^d - 1$ [24].

Two alternative algorithms have been proposed: Broadcast Incremental Power (BIP) [42] and Adaptive Broadcast Consumption (ABC) [30]. BIP repeatedly adds an edge to an uncovered node that adds least to the current range assignment starting with the root node. ABC finds the cheapest way to add the node that has the smallest distance to the current set of covered nodes. The cost is computed by considering for every node a range increase that would cover the new node and from which subsequently the ranges that are now superfluous have been removed.

Both algorithms are at least as good as MST [41, 30]. For BIP a lower bound of $13/3$ is known [41] and for ABC a lower bound of 2. By using an integer linear program formulation for the optimum, the three approximation algorithms have been compared experimentally with the optimum [30]. The experiments show that ABC outperforms BIP on average, but not always, and both outperform MST consistently. In the experiments over hundreds of instances even MST never exceeded over more than a factor 2 of the optimum. So one must be really "lucky" if an instance has an MST-solution 6 times worse than the optimum.

Recently a new approximation algorithm has been proposed with an approximation ratio of $2 \ln \rho - 2 \ln 2 + 2$ if $\rho > 2$ and ρ if $\rho \leq 2$ where ρ is the approximation ratio of the MST-heuristic [11]. So the ratio is improved to 4.2 for $d = 2$, which is better than the lower bound for BIP, and to 6.49 for $d = 3$. The algorithm works by finding good *contractions*: Repeatedly improve a spanning tree by increasing a range and removing the longest edge in each cycle.

2.5 Bounded hop

For $h = 1$ the broadcast problem is trivial. The problem has also been solved optimally for $h = 2$ in the plane with time complexity $O(n^7)$ [3]. The same paper also gives a PTAS in the plane with time complexity $O(n^\mu)$ where $\mu = O\left((\alpha 2^\alpha h^\alpha / \epsilon)^{\alpha^h}\right)$. Recently this has been improved to $O\left(n + \left(\frac{4h}{\epsilon}\right)^{\left(\frac{1}{\epsilon}\right)^{4h}}\right)$ for a $(1 + (\alpha + 2)\epsilon)$ -approximation by reducing any

instance to a simpler problem, called a *coreset*, of size $O((\frac{1}{\epsilon})^{4h})$ [26]. Note that the size of the coreset does not depend on n and the running time of the algorithm is only linear in n .

For hSC a constant-factor approximation exists. The approximation factor, however, is $(1/(\sqrt[h]{2} - 1))^\alpha(1 + 3^\alpha)(2 * 3^\alpha)^{h-2}$ and the algorithm consists of multiple approximation reserving reductions [27], so it's not really practical. A more practical approach has been taken by Clementi et al. [17], who prove a lower and an upper bound on the optimum in the plane, where the latter is obtained from a simple divide-and-conquer algorithm. For well-spread instances the upper-bound matches the lower-bound and hence the algorithm yields a constant-factor approximation for any family of well-spread instances (but with unknown constant). Another practical heuristic has been proposed for the plane that works by greedily lowering ranges starting from a minimal range assignment where all stations are required to send with the same range, but no analytical results have been provided [21].

2.6 Heterogeneous networks

In a heterogeneous network the cost to reach a node t from s can be arbitrary instead of $|st|^\alpha$. It is modeled by assigning a weight $c(s, t)$ to every pair of nodes $\{s, t\}$ such that s can reach t if $r(s) \geq c(s, t)$. The MST algorithm still yields a 2-approximation for SC and C if the cost function is symmetric: $c(s, t) = c(t, s)$ [1]. In general, however, the MST-algorithm does not yield a 2-approximation [4]. Another algorithm, however, does yield a 2-approximation: repeatedly select for every node the minimum weighted outgoing arc and subsequently contract connected components into supernodes [23]. For B the problem can't be approximated within a sub-logarithmic factor unless $P=NP$ [14] and a $(2 + 2 \ln(n - 1))$ -approximation algorithm has been developed [10].

2.7 Directional Antennas

Very recently Caragiannis et al. [12] investigate a problem similar to ours. They study the problem of finding a minimum range and orientation of a set of directed antennas such that the resulting communication graph is strongly connected. All stations transmit with the same range and with the same angular spread. Hence the problem they study differs with ours in that we allow all stations to transmit with a different range. They give a $2 \sin(\pi - \phi/2)$ -approximation for $\pi \leq \phi \leq 8\pi/5$ where ϕ denotes the angle of the antenna's beam and a 3-approximation for $\phi < \pi$.

Chapter 3

Approach

In this chapter we outline the approach we've taken on finding constant-factor approximations for the dSC problem. First of all note that the most important results in the literature use the MST as a basis. Kirousis et al. [29] have shown that the MST heuristic is a 2-approximation for SC. Let's look at the proof. Let $\text{cost}(r_{\text{OPT}})$ denote the cost of the optimal range assignment and $\text{weight}(\text{MST}) = \sum_{\{u,v\} \in \text{MST}} |uv|^\alpha$ the total weight of the MST. We need the following lemma.

Lemma 3.1 ([29]). *The MST is a lower bound for the SC problem: $\text{cost}(r_{\text{OPT}}) \geq \text{weight}(\text{MST})$.*

Proof. The communication graph of the optimum is a strongly connected (directed) graph. This graph must contain at least a directed spanning tree T directed towards a station s because all nodes can reach s . In this tree each node except the root has one edge leading towards the root. So we can associate the cost $r^\alpha(v)$ of each station v with that edge. Let $\text{weight}(T)$ be the sum of all the edge weights of T . The cost of the optimum solution is at least $\text{weight}(T)$ because the range of the root node is not included. And the weight of T is at least the weight of the Minimum (undirected) Spanning Tree. So we have $\text{cost}(r_{\text{OPT}}) \geq \text{weight}(T) \geq \text{weight}(\text{MST})$. \square

We note that same argument also applies for dSC.

Lemma 3.2. *The MST is a lower bound for the dSC problem: $\text{cost}(r_{\text{OPT}}) \geq \text{weight}(\text{MST})$.*

Theorem 3.3 ([29]). *The MST-heuristic is a 2-approximation for the SC problem.*

Proof. We will denote the range assignment of the algorithm with r_{MST} . We have to show that $\text{cost}(r_{\text{MST}}) \leq 2 \cdot \text{cost}(r_{\text{OPT}})$. By Lemma 3.1 we have $\text{weight}(\text{MST}) \leq \text{cost}(r_{\text{OPT}})$ so all we need to show is that $\text{cost}(r_{\text{MST}}) \leq 2 \cdot \text{weight}(\text{MST})$.

In the MST-heuristic the range of a node v is set to the maximum weight of all edges of the MST that are incident to v . Recall that the weight of an edge $\{u, v\}$ is set to the Euclidean

distance between u and v to the power α . We denote this with $|uv|^\alpha$. So we have:

$$\text{cost}(r_{\text{MST}}) = \sum_{v \in S} \max_{\{u,v\} \in \text{MST}} |uv|^\alpha \quad (3.1)$$

$$\leq \sum_{v \in S} \sum_{\{u,v\} \in \text{MST}} |uv|^\alpha \quad (3.2)$$

$$= 2 \sum_{\{u,v\} \in \text{MST}} |uv|^\alpha \quad (3.3)$$

$$= 2 \cdot \text{weight}(\text{MST}) \quad (3.4)$$

Inequality (3.3) follows from (3.2) because in (3.2) every edge of the MST is counted exactly twice. \square

If the angle of the directional antennas are really small and no three points are collinear, then the best we can hope for is a tour through all the nodes of minimal weight: a traveling salesman tour (TSP). Therefore we define the following TSP-problem.

Definition 3.4. We define TSP^α to be the TSP-problem where the cities are given as a set of points P in an Euclidean space and the weight $w(u, v)$ of every edge $\{u, v\} \in P^2$ is given by $w(u, v) = |uv|^\alpha$. We denote the weight of a tour T with $\text{weight}(T) = \sum_{\{u,v\} \in T} |uv|^\alpha$.

There exist some algorithms that are a 2-approximation for geometric TSP and also use the MST as a basis. They cannot be applied in a straight-forward manner because the edge weights do not satisfy the triangle inequality. But they do also have the MST as a lower bound. This result extends to TSP^α . We denote the optimal tour for TSP^α with T_{OPT} .

Lemma 3.5. *The MST is a lower bound for TSP^α : $\text{weight}(\text{MST}) \geq \text{weight}(T_{\text{OPT}})$.*

Proof. If we remove one edge from the optimal tour T_{OPT} the result is a simple path. The simple path is a spanning tree of the input set so its weight is at least the weight of the minimum spanning tree. \square

In the next two chapters we will show that two 2-approximations for geometric TSP also yield a constant factor approximation for TSP^α using Lemma 3.5. By Lemma 3.2 these results imply the same approximation factors for the dSC problem.

Chapter 4

The T^3 -algorithm

In the previous chapter we've argued that approximation algorithms that are constant-factor approximations for Geometric TSP and use the MST as a lower bound might also be constant-factor approximations for the dSC and TSP^α problems. In this chapter we first show that this is true for the T^3 -algorithm. Then we further improve the approximation factor by changing the algorithm.

4.1 Relation with relaxed TSP

The T^3 -algorithm by Andreae and Bandelt [7] is designed for the case that the TSP input satisfies a relaxed triangle inequality.

Definition 4.1. Let $G = (V, E; w)$ be a complete weighted graph with weight function $w : E \rightarrow \mathbb{R}_{\geq 0}$. Then we say that G fulfills a parameterized triangle inequality if for any triangle Δabc it holds that:

$$w(a, c) \leq \tau(w(a, b) + w(b, c))$$

The inequality is called *relaxed* if $\tau > 1$ and it is called *sharpened* if $\frac{1}{2} \leq \tau < 1$.

The problem of finding a traveling salesman tour in G if G fulfills a relaxed triangle inequality is denoted with Δ_τ TSP.

The factor τ is always at least $1/2$ [7]. In our case the graph satisfies a relaxed triangle inequality as the next lemma shows. But first recall Hölders inequality.

Theorem 4.2 (Hölders inequality). *If $p, q > 1$ and $\frac{1}{p} + \frac{1}{q} = 1$ then for $(x_1, \dots, x_n), (y_1, \dots, y_n) \in \mathbb{R}^n$*

$$\sum_{i=1}^n |x_i y_i| \leq \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} \left(\sum_{i=1}^n |y_i|^q \right)^{1/q}$$

Lemma 4.3. *Let S be a set of points in an m -dimensional Euclidean space and let $|st|$ denote the distance between points s and t . Then the complete weighted graph $G = (S, S^2; w)$ with weight function given by $w(s, t) = |st|^\alpha$ fulfills a relaxed triangle inequality with $\tau = 2^{\alpha-1}$.*

Proof. Let a , b and c be the Euclidean distances of the edges of a triangle in this graph. Then we need to prove that:

$$a^\alpha \leq 2^{\alpha-1}(b^\alpha + c^\alpha). \quad (4.1)$$

Using Hölder's inequality with $p = \alpha$, $q = \frac{\alpha}{\alpha-1}$, $n = 2$, $x_1 = b$, $x_2 = c$ and $y_1 = y_2 = 1$ and because the distances b and c are always non-negative we obtain:

$$b + c \leq (b^\alpha + c^\alpha)^{1/\alpha} \cdot 2^{(\alpha-1)/\alpha}.$$

And thus:

$$(b + c)^\alpha \leq 2^{\alpha-1}(b^\alpha + c^\alpha). \quad (4.2)$$

The (normal) triangle inequality states that $a \leq b + c$. So we also have

$$a^\alpha \leq (b + c)^\alpha \quad (4.3)$$

(4.2) and (4.3) together imply (4.1). Inequality (4.1) is tight if $a = b + c$ and $b = c$. \square

Andreae and Bandelt [7] give an approximation algorithm for Δ_τ TSP. Their T^3 -algorithm is an adaptation of the well-known double-spanning-tree heuristic for TSP: find a minimum spanning tree (MST) in G , double all the edges of the MST, construct an Euler tour in the resulting multigraph and construct a Hamiltonian cycle from this Euler tour by skipping all nodes that have already been visited. The heuristic is a 2-approximation if the triangle inequality holds because then skipping over visited nodes never increases the length of the Euler tour which is exactly twice the weight of the MST. The weight of the MST is a lower bound for TSP, because removing any edge from the optimal tour yields a spanning tree whose weight is at least the weight of the minimum spanning tree.

The T^3 -algorithm of Andreae and Bandelt also creates a Hamiltonian tour by shortcutting the MST, but their algorithm never skips more than two consecutive nodes. It is never necessary to skip more than two consecutive nodes because the cube T^3 of a tree T is always Hamiltonian by a result of Sekanina [40]. The cube of a graph G contains an edge $\{u, v\}$ if there is a path from u to v in G that uses at most three edges. The proof of Sekanina is constructive, and Andreae and Bandelt use it to construct a tour in MST^3 .

The following is the recursive procedure of Sekanina [40] to obtain a Hamiltonian Tour in T^3 . We denote the number of nodes in a tree T with $|T|$.

Algorithm $HCT^3(T, b)$

Input: A tree T with $|T| \geq 3$ and an edge $b = \{b_1, b_2\}$ of T .

Output: A Hamiltonian cycle of T^3 containing b .

1. **for** $i \leftarrow 1$ **to** 2
2. **do** $T_i \leftarrow$ component from $T - b$ that contains b_i
3. (* construct tour for T_i *)
4. **if** $|T_i| = 1$
5. **then** $P_i \leftarrow \emptyset$
6. $v_i \leftarrow b_i$
7. **else** Pick an edge $\{v_i, b_i\}$ of T_i
8. (* The Hamiltonian tour of T_i consists of the path P_i and the edge $\{v_i, b_i\}$ *)
9. **if** $|T_i| = 2$
10. **then** $P_i \leftarrow \{v_i, b_i\}$
11. **else** $P_i \leftarrow HCT^3(T_i, \{v_i, b_i\}) - \{v_i, b_i\}$
12. **return** The Hamiltonian tour that consists of P_1, b, P_2 and the shortcut $\{v_1, v_2\}$

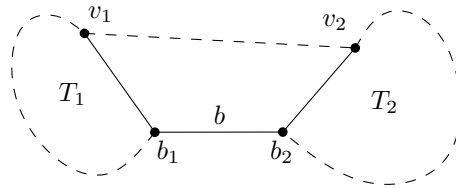


Figure 4.1: Intuitive working of the HCT^3 -procedure

Figure 4.1 illustrates the working of the HCT^3 -procedure. The algorithm is applied to a tree T and an edge b of T with endpoints b_1 and b_2 . Removal of the edge b splits the tree into two components T_1 and T_2 . In both components the algorithm selects an edge $\{v_i, b_i\}$ that is incident to b and applies the algorithm recursively to obtain an Hamiltonian cycle of T_i that includes the edge $\{v_i, b_i\}$. In the next step the two cycles are merged to form one Hamiltonian cycle of T that includes edge b . In the merging step the algorithm removes the edges $\{v_i, b_i\}$ from the two cycles and adds the edges b and $\{v_1, v_2\}$.

The T^3 -algorithm applies this procedure to a minimum spanning tree of the input graph. We consider three versions of the T^3 -algorithm.

- The *standard* T^3 -algorithm.
- The *refined* T^3 -algorithm of Andreae [6].
- The *geometric* T^3 -algorithm.

The refined T^3 -algorithm makes a better choice for the edge $\{v_i, b_i\}$ on line 7. For our analysis we will also make a different choice here. We define the geometric version in the next section.

The T^3 -algorithm has a performance guarantee of $3\tau^2/2 + \tau/2$. Let H be the resulting tour. Then more precisely Andreae and Bandelt prove that $\text{weight}(H) \leq (3\tau^2/2 + \tau/2)\text{weight}(\text{MST})$. Andreae [6] improves the algorithm by finding a better Hamiltonian tour in MST^3 with weight at most $(\tau^2 + \tau)\text{weight}(\text{MST})$. For the SC problem (with or without directional antenna) we also have the same lower bound $\text{weight}(r_{\text{OPT}}) \geq \text{weight}(\text{MST})$ by Lemma 3.2. Hence the refined T^3 -algorithm is also an approximation algorithm for SC and dSC with $\tau = 2^{\alpha-1}$.

Corollary 4.4. *The refined T^3 -algorithm is a $(4^{\alpha-1} + 2^{\alpha-1})$ -approximation for TSP^α and (d)SC.*

Carmi et al. [13] already prove that the refined T^3 -algorithm is a 6-approximation for $\alpha = 2$.

We can improve the analysis a little using a simpler argument. We'll frequently use the following definition.

Definition 4.5. A k -shortcut of a tree T is an edge $\{v_0, v_k\}$ where v_0, \dots, v_k is a simple path p in T . We say that a k -shortcut *uses* an edge e if e is on the path p .

By Lemma 4.3 the weight of a 2-shortcut that uses edges a and b is at most $2^{\alpha-1}(|a|^\alpha + |b|^\alpha)$. This result can be generalized to k -shortcuts. The proof is given in the appendix on page 63.

Lemma 4.6. *Let e be a k -shortcut that uses edges e_1, \dots, e_k . Then the following inequality holds:*

$$|e|^\alpha \leq k^{\alpha-1} \sum_{1 \leq i \leq k} |e_i|^\alpha$$

The tour constructed by the T^3 -algorithm consists of edges of T and 2- and 3-shortcuts that use edges of T . In this tour each edge of T is used exactly twice. Thus the original T^3 -algorithm already does better than that Corollary 4.4 states for the refined T^3 -algorithm.

Corollary 4.7. *Every version of the T^3 -algorithm is a $2 \cdot 3^{\alpha-1}$ -approximation for (d)SC and TSP^α .*

Andreae and Bandelt [7] give an example graph for which any Hamiltonian tour that uses only edges of MST^3 has a length of at least $(\tau^2 + \tau)\text{weight}(\text{MST})$. So the T^3 -algorithm cannot be improved by finding a better tour in MST^3 . It can easily be verified that the square of a tree is not always Hamiltonian. See for example Figure 4.2. If we allow m -shortcuts for some $m > 3$, it would lead to skipping more nodes in the Hamiltonian cycle and hence increase the approximation factor to $O(\tau^m)$. It is unlikely that the approximation

factor of Δ_τ TSP can be improved using (only) the MST lower bound. Indeed Bender and Chekuri [9] design an improved 4τ -approximation using a different lower bound: the optimal TSP tour is a 2-node connected subgraph of the original graph. The weight of the optimal TSP tour is at least that of the minimum-weight 2-node-connected subgraph. The latter is NP-hard to compute, but can be approximated within a factor 2 [28]. Furthermore the square of a 2-node-connected subgraph is always Hamiltonian. Thus using only edges of the subgraph and 2-shortcuts yields an $O(\tau)$ -approximation. We cannot use this algorithm for SC or dSC because the lower bound does not need to hold: the communication graph of an optimal solution for SC could be a tree with bidirected edges for example. For TSP $^\alpha$, however, Lemma 4.3 immediately implies that their algorithm is a $2^{\alpha+1}$ -approximation.

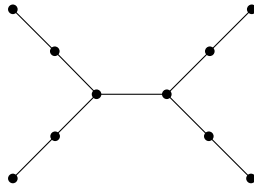


Figure 4.2: Example showing a (minimum spanning) tree whose square is not Hamiltonian.

4.2 Improved analysis of the T^3 -algorithm for points in the plane and $\alpha = 2$

Until now we've used only graph-theoretic arguments to determine the approximation factor of the T^3 -algorithm for TSP $^\alpha$. The analysis can be improved if we restrict ourselves to points in the plane. In the previous analysis of the T^3 -algorithm we've exploited that each edge is used in two (≤ 3)-shortcuts. The weight of a 3-shortcut is maximum if the three points lie on a line. In case of the Euclidean MST we know that two edges of the MST that share an endpoint make an angle of at least $\pi/3$. This property also holds for the MST with power-weighted edges as shown in the appendix, Lemma A.1. Because these edges must make an angle, it is not possible for all points to lie on a straight line unless the degree of the points is at most two. If, however, the degree is at most two, the T^3 -algorithm generates a 2-shortcut only at one side. So the idea to improve the analysis of the T^3 -algorithm is to exploit the following: depending on the degree of the endpoints either an edge cannot be used in two 3-shortcuts or the weight of the 3-shortcuts is less than previously argued. To stimulate the generation of cheaper shortcuts we study the following variant of the T^3 -algorithm.

Definition 4.8. The *geometric* T^3 -algorithm is a variant of the T^3 -algorithm that chooses on line 7 the edge $\{v_i, b_i\}$ that makes the smallest angle with b .

In this section we only consider the case $\alpha = 2$. In the next section we will further improve the upper bound on the approximation factor and in Section 4.4 we show how the results

generalize to all values $\alpha \geq 2$. We first derive an expression for the length of a 3-shortcut. We need the following definitions.

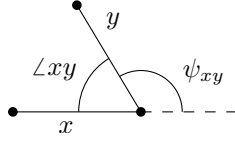
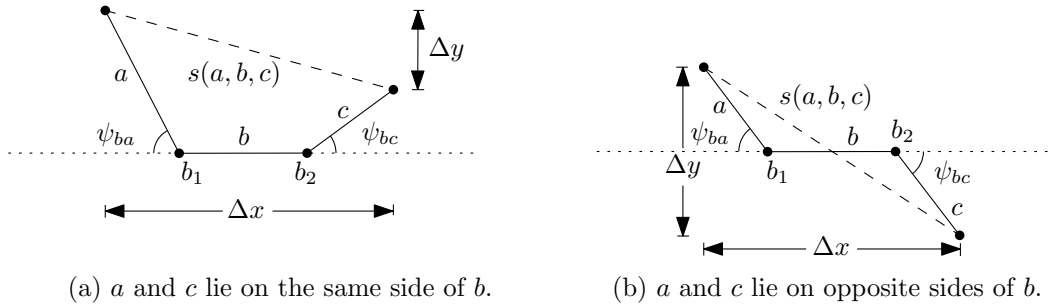


Figure 4.3: The angle ψ_{xy} is defined to be the angle between y and the extension of x .

Definition 4.9. When edges x and y share an endpoint we denote the angle between y and the extension of x with ψ_{xy} . So $\psi_{xy} = \pi - \angle xy$. See Figure 4.3.

Definition 4.10. If $a = \{v_1, v_2\}$, $b = \{v_2, v_3\}$ and $c = \{v_3, v_4\}$ are three consecutive edges then we define $s(a, b, c)$ to be the 3-shortcut $\{v_1, v_4\}$.



(a) a and c lie on the same side of b .

(b) a and c lie on opposite sides of b .

Figure 4.4: Two cases for computing the length of a 3-shortcut $s(a, b, c)$.

Lemma 4.11. *The weight of a 3-shortcut $s(a, b, c)$ is given by:*

$$|s(a, b, c)|^2 = |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \cos(\psi_{ba} \pm \psi_{bc})$$

where $\pm = \begin{cases} + & \text{if } a \text{ and } c \text{ lie on the same side of the line through } b, \\ - & \text{if } a \text{ and } c \text{ lie on opposite sides.} \end{cases}$

(4.4)

Figure 4.4 shows the two cases.

Proof. First suppose a and c lie on the same side.

$$\begin{aligned}
|s(a, b, c)|^2 &= (\Delta x)^2 + (\Delta y)^2 \\
&= (|a| \cos \psi_{ba} + |b| + |c| \cos \psi_{bc})^2 + ||a| \sin \psi_{ba} - |c| \sin \psi_{bc}|^2 \\
&= (|a| \cos \psi_{ba} + |b| + |c| \cos \psi_{bc})^2 + (|a| \sin \psi_{ba} - |c| \sin \psi_{bc})^2 \\
&= |a|^2 (\cos^2 \psi_{ba} + \sin^2 \psi_{ba}) + |b|^2 + |c|^2 (\cos^2 \psi_{bc} + \sin^2 \psi_{bc}) \\
&\quad + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| (\cos \psi_{ba} \cos \psi_{bc} - \sin \psi_{ba} \sin \psi_{bc}) \\
&= |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \cos(\psi_{ba} + \psi_{bc})
\end{aligned}$$

Similarly, if a and c lie on opposite sides:

$$\begin{aligned}
|s(a, b, c)|^2 &= (\Delta x)^2 + (\Delta y)^2 \\
&= (|a| \cos \psi_{ba} + |b| + |c| \cos \psi_{bc})^2 + (|a| \sin \psi_{ba} + |c| \sin \psi_{bc})^2 \\
&= |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \cos(\psi_{ba} - \psi_{bc})
\end{aligned}$$

□

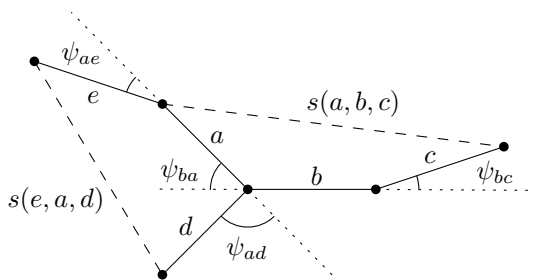


Figure 4.5: Two 3-shortcuts that use the edge a .

In Lemma 4.11 we express the weight of a 3-shortcut in terms of the lengths of the edges and the angles between them. Now we show that if an edge a is used in two 3-shortcuts two of those angles are related. First note that in this case the T^3 -algorithm generates the two 3-shortcuts in two consecutive recursive calls. See Figure 4.5. The T^3 -algorithm is first applied to edge b and recursively to edge a . In the recursive call the shortcut $s(e, a, d)$ is generated that uses edges e , a and d where a is the middle edge and d is the edge that is incident to both a and b . Then the algorithm returns from the recursion and generates the 3-shortcut $s(a, b, c)$ that uses edges a , b and c where b is the middle edge. So by the way the T^3 -algorithm generates these shortcuts it must be the case that in one 3-shortcut a is the middle edge and in the other 3-shortcut a is either the first or the third edge.

Lemma 4.12. *If the geometric T^3 -algorithm generates in two recursive calls the two 3-shortcuts $s(a, b, c)$ and $s(e, a, d)$ such that b is the middle edge of the first 3-shortcut and a*

is the middle edge of the second 3-shortcut and d is incident to both a and b , then we have that:

$$\psi_{ba} \geq \frac{\pi - \psi_{ad}}{2}.$$

Proof. The geometric T^3 -algorithm is first applied to edge b and recursively to edge a . We choose to proceed with the edge that makes the smallest angle with b , so we know that $\angle ab \leq \angle bd$ and it follows that $\psi_{ba} \geq \psi_{bd}$. Now we can relate ψ_{ba} with the angle ψ_{ad} .

$$\begin{aligned} \psi_{ba} &\geq \frac{\psi_{ba} + \psi_{bd}}{2} \\ &= \frac{\angle ad}{2} \\ &= \frac{\pi - \psi_{ad}}{2} \end{aligned}$$

□

Now we're ready to prove the main theorem of this section.

Theorem 4.13. *The geometric T^3 -algorithm is a $5\frac{1}{8}$ -approximation for TSP^2 .*

Proof. We know that each edge is used in two (≤ 3)-shortcuts. By Lemma 4.6 we know that we can upper bound the weight of a shortcut that uses edges e_1, \dots, e_k as a linear combination of the weights of the edges: $|e_1 e_k|^2 \leq c_1 \cdot |e_1|^2 + \dots + c_k \cdot |e_k|^2$. The weight of the tour is the sum of the weights of all shortcuts. For a given edge a we can take the two terms in this sum that contain a together. We say that this value is the *contribution* of a to the tour. In the sequel we will denote the contribution of a by $\text{contribution}(a)$. We show that the contribution of each edge a is at most $5\frac{1}{8}|a|^2$. The theorem then follows by summing up the contribution of all edges. We consider two cases.

Case I: The edge a is used in a (≤ 2)-shortcut on one side and a (≤ 3)-shortcut on the other side.

The total contribution of a to the tour is at most $5|a|^2$ by Lemma 4.6.

Case II: The edge a is used in two 3-shortcuts.

By Lemma 4.11 we know that we can express the weight of a 3-shortcut $s(a, b, c)$ by equation (4.4). To further simplify the expression we rewrite the composite terms using Young's inequality. For two real numbers x and y , Young's inequality states that

$$xy \leq \frac{x^2}{2} + \frac{y^2}{2}.$$

This fact follows from observing that

$$(x - y)^2 \geq 0.$$

We use Young's inequality on all three composite terms in equation (4.4).

$$\begin{aligned}
|s(a, b, c)|^2 &= |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \cos(\psi_{ba} \pm \psi_{bc}) \\
&\leq |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \\
&\leq |a|^2 + |b|^2 + |c|^2 + (|a|^2 + |b|^2) \cos \psi_{ba} + (|b|^2 + |c|^2) \cos \psi_{bc} + (|a|^2 + |c|^2) \\
&= (2 + \cos \psi_{ba})|a|^2 + (1 + \cos \psi_{ba} + \cos \psi_{bc})|b|^2 + (2 + \cos \psi_{bc})|c|^2 \quad (4.5)
\end{aligned}$$

Now we return to the case that the edge a is used in two 3-shortcuts $s(a, b, c)$ and $s(e, a, d)$ as in Figure 4.5. In the first shortcut a must be the first or the third edge and in the second one a must be the middle edge. We will also assume that the edge d is incident to both a and b . The weight of the final tour is the sum of the weights of all shortcuts. We bound the weight of all 3-shortcuts by inequality (4.5). Then we can take the terms that contain a together:

$$\text{contribution}(a) = (3 + \cos \psi_{ba} + \cos \psi_{ae} + \cos \psi_{ad})|a|^2 \leq (4 + \cos \psi_{ba} + \cos \psi_{ad})|a|^2 \quad (4.6)$$

Note that to obtain inequality (4.6) it doesn't matter whether a is the first or the last edge used in $s(a, b, c)$ since inequality (4.5) is symmetric in a and c . We can upper bound the term $\cos \psi_{ba} + \cos \psi_{ad}$ in inequality (4.6) because by Lemma 4.12 we have $\psi_{ba} \geq (\pi - \psi_{ad})/2$. Since we also have $\psi_{ba} \leq \pi$ it follows that $\cos \psi_{ba} \leq \cos((\pi - \psi_{ad})/2) = \sin(\psi_{ad}/2)$. Now we can compute the maximum of $f(\psi_{ad}) = \cos \psi_{ad} + \sin(\psi_{ad}/2)$ with $0 \leq \psi_{ad} \leq \pi$. Equating the derivative to zero yields:

$$\begin{aligned}
f'(\psi_{ad}) &= -\sin \psi_{ad} + \frac{1}{2} \cos \frac{\psi_{ad}}{2} = 0 \\
2 \sin \frac{\psi_{ad}}{2} \cdot \cos \frac{\psi_{ad}}{2} &= \frac{1}{2} \cos \frac{\psi_{ad}}{2}
\end{aligned}$$

We consider two cases. Case (a): $\cos \frac{\psi_{ad}}{2} = 0$. Then $\psi_{ad} = \pi$. Case (b): $\cos \frac{\psi_{ad}}{2} \neq 0$. Then

$$\begin{aligned}
\sin \frac{\psi_{ad}}{2} &= \frac{1}{4} \\
\psi_{ad} &= 2 \sin^{-1} \frac{1}{4}
\end{aligned}$$

The second-derivative test shows that the function has a minimum at $\psi_{ad} = \pi$ and a maximum at $\psi_{ad} = 2 \sin^{-1}(1/4)$. Now compute the maximum:

$$\begin{aligned}
\cos \psi_{ad} + \sin \frac{\psi_{ad}}{2} &\leq \cos \left(2 \sin^{-1} \frac{1}{4} \right) + \frac{1}{4} \\
&= 1 - 2 \sin^2 \left(\sin^{-1} \frac{1}{4} \right) + \frac{1}{4} \\
&= \frac{1}{8}
\end{aligned}$$

Plugging this result in (4.6) using $\cos \psi_{ba} \leq \sin(\psi_{ad}/2)$ we obtain $\text{contribution}(a) \leq 5\frac{1}{8}|a|^2$. Recall that in case I the contribution of a is at most $5|a|^2$. So in both cases the contribution of any edge a to the tour is at most $5\frac{1}{8}|a|^2$. The theorem follows by summing up the contribution of all edges of the MST. \square

4.3 Further improvement for points in the plane and $\alpha = 2$

In Theorem 4.13 we've bounded the contribution of one edge to the tour. We've seen that in the worst case an edge is used in two 3-shortcuts, but due to our smallest-angle selection rule, the two 3-shortcuts cannot be of maximal length at the same time. That is because, by Lemma 4.11, the length of a 3-shortcut depends on the angles between the three edges. Recall that by Lemma 4.11 the weight of a 3-shortcut $s(a, b, c)$ is given by:

$$|s(a, b, c)|^2 = |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \cos(\psi_{ba} \pm \psi_{bc}).$$

Subsequently we've used Young's inequality to rewrite all composite terms in (4.3). In the next theorem we rewrite the composite terms differently using Young's inequality with ϵ . For two real numbers x and y and for $\epsilon > 0$, Young's inequality with ϵ states that

$$xy \leq \frac{x^2}{2\epsilon} + \frac{y^2\epsilon}{2}.$$

This fact follows from observing that

$$\left(\frac{x}{\sqrt{\epsilon}} - y \cdot \sqrt{\epsilon} \right)^2 \geq 0.$$

Young's inequality with ϵ allows us to charge one edge of a composite term more than the other. Why would this help us? This helps because by the topology of the MST in the plane there can only be a limited number of edges adjacent to the same vertex, and the more vertices there are, the smaller the angles between them become. To give some intuition, see Figure 4.6.

In Figure 4.6 the edge a is used in two 3-shortcuts. In the weight of the 3-shortcut $s(a, b, c)$ we get the composite term $2|a||b| \cos \psi_{ba}$. In Theorem 4.13 we've upper bounded this term by $(|a|^2 + |b|^2) \cos \psi_{ba}$. When we consider the contribution of b , this information is only useful if there is another 3-shortcut that uses b and another edge incident to the same vertex as a and b as the middle edge. For example in Figure 4.6 we've drawn a 3-shortcut $s(b, f, g)$ where f is the middle edge. If such a shortcut does not exist then we would only use that $\cos \psi_{ba}|b|^2 \leq |b|^2$ while upper-bounding the contribution of b . So in that case we might as well apply Young's inequality asymmetrically and charge more to edge b and less to edge a . Then we get: $2|a||b| \cos \psi_{ba} \leq |a|^2 \cos^2 \psi_{ba} + |b|^2$ by taking $\epsilon = 1/\cos \psi_{ba}$ if $\cos \psi_{ba} > 0$. Using Young's inequality with ϵ and a more extensive case analysis we can further improve the approximation factor.

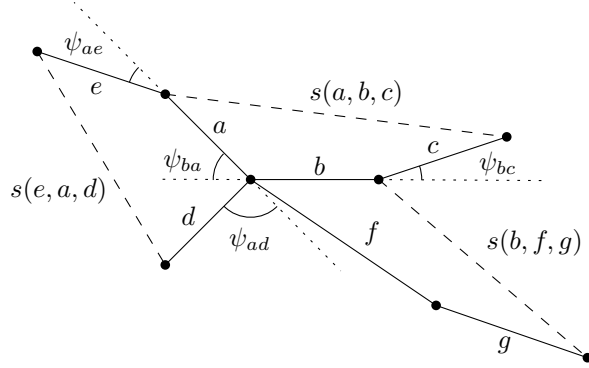


Figure 4.6: Two 3-shortcuts that use the edge a and two 3-shortcuts that use the edge b .

Theorem 4.14. *The geometric T^3 -algorithm is a 5-approximation for TSP^2 .*

Proof. As in the proof of Theorem 4.13 we will bound the contribution of each edge a to the tour. In the proof of Theorem 4.13 we've already seen that the contribution is at most $5|a|^2$ when a is used in a (≤ 2)-shortcut on one side and a (≤ 3)-shortcut on the other side. So we reconsider the case that a is used in two 3-shortcuts, see Figure 4.7. By Lemma 4.11 we can express the weight of a 3-shortcut $s(a, b, c)$ as follows.

$$\begin{aligned}
|s(a, b, c)|^2 &= |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \cos(\psi_{ba} \pm \psi_{bc}) \\
&\leq |a|^2 + |b|^2 + |c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} + 2|a||c| \\
&\leq 2|a|^2 + |b|^2 + 2|c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc}
\end{aligned} \tag{4.7}$$

We rewrite the composite terms in (4.7) using Young's inequality with ϵ . Let v be the vertex that is incident to edges a and b . If there are multiple 3-shortcuts that use edges that are incident to v then the T^3 -algorithm generates these in consecutive recursive calls. We renumber the edges incident to v by the level of recursion in which the algorithm is applied to an edge incident to v . Of all edges incident to v let the algorithm first be applied to the edge $\{v, v_1\}$, then recursively to $\{v, v_2\}$ etc. Then we have $b = \{v, v_i\}$ and $a = \{v, v_{i+1}\}$ for some $i \geq 1$, because the algorithm is first applied to b and then recursively to a . We define $\psi_i = \psi_{ba} = \psi_{\{v, v_i\}\{v, v_{i+1}\}}$. We rewrite the term $2|a||b| \cos \psi_{ba}$ in (4.7) as follows.

$$2|a||b| \cos \psi_{ba} = 2|vv_i||vv_{i+1}| \cos \psi_i \leq f(|vv_{i+1}|, |vv_i|, \psi_i), \tag{4.8}$$

$$\text{where } f(|vv_{i+1}|, |vv_i|, \psi_i) = \begin{cases} 0 & \text{if } \psi_i \geq \frac{\pi}{2}, \\ |vv_i|^2 + |vv_{i+1}|^2 \cos^2 \psi_i & \text{if } \psi_i < \frac{\pi}{2} \text{ and} \\ & (i = 1 \text{ or } (i > 1 \text{ and } \psi_{i-1} \geq \frac{\pi}{2})), \\ (|vv_i|^2 + |vv_{i+1}|^2) \cos \psi_i & \text{if } \psi_i < \frac{\pi}{2} \text{ and } i > 1 \text{ and } \psi_{i-1} < \frac{\pi}{2}. \end{cases}$$

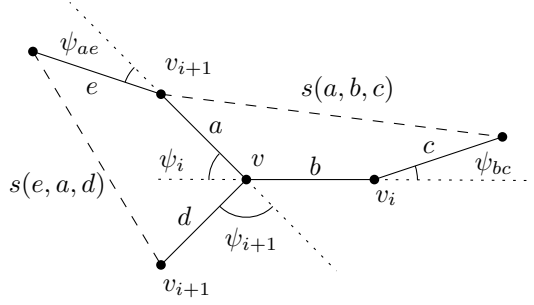


Figure 4.7: Two 3-shortcuts that use the edge a

The second case of inequality (4.8) follows from Young's inequality with $\epsilon = 1/\cos \psi_i$ and the third case from Young's inequality with $\epsilon = 1$. The term $2|b||c| \cos \psi_{bc}$ is replaced similarly in (4.7) so that we get the following inequality.

$$\begin{aligned} |s(a, b, c)|^2 &\leq 2|a|^2 + |b|^2 + 2|c|^2 + 2|a||b| \cos \psi_{ba} + 2|b||c| \cos \psi_{bc} \\ &\leq 2|a|^2 + |b|^2 + 2|c|^2 + f(|a|, |b|, \psi_{ba}) + f(|c|, |b|, \psi_{bc}) \end{aligned} \quad (4.9)$$

We use Inequality (4.9) to bound the weights of all 3-shortcuts. The weight of the final tour is the sum of the weights of all shortcuts. In this sum we can take the two occurrences of an edge $a = \{v, v_{i+1}\}$ together and analyze the contribution of a to the tour. Note that the result of (4.9) is still at most $3(|a|^2 + |b|^2 + |c|^2)$. So if an edge a is used in a (≤ 3)-shortcut on one side and a (≤ 2)-shortcut on the other side, then we still have that the contribution of a is at most $5|a|^2$. It remains to consider the case that an edge a is used in two 3-shortcuts. Let $s(a, b, c)$ and $s(e, a, d)$ be these 3-shortcuts. The algorithm is first applied to edge b and generates shortcut $s(a, b, c)$, where a is the first or the third edge of the shortcut. Then the algorithm is recursively applied to edge a and generates shortcut $s(e, a, d)$, where a is the middle edge. Figure 4.7 shows how the vertices are numbered in this case.

Let π_a be a function that takes a sum of terms and returns the sum of all terms that contain $|a|$. We derive the following expression for the contribution of a .

$$\begin{aligned} \text{contribution}(a) &= \pi_a(\text{weight}(s(a, b, c))) + \pi_a(\text{weight}(s(e, a, d))) \\ &\leq \pi_a(2|a|^2 + |b|^2 + 2|c|^2 + f(|a|, |b|, \psi_{ba}) + f(|c|, |b|, \psi_{bc})) \\ &\quad + \pi_a(2|e|^2 + |a|^2 + 2|d|^2 + f(|e|, |a|, \psi_{ae}) + f(|d|, |a|, \psi_{ad})) \\ &= 3|a|^2 + \pi_a(f(|a|, |b|, \psi_{ba})) + \pi_a(f(|e|, |a|, \psi_{ae})) + \pi_a(f(|d|, |a|, \psi_{ad})) \\ &\leq 4|a|^2 + \pi_a(f(|a|, |b|, \psi_{ba})) + \pi_a(f(|d|, |a|, \psi_{ad})) \\ &= 4|a|^2 + \pi_a(f(|vv_{i+1}|, |vv_i|, \psi_i)) + \pi_a(f(|vv_{i+2}|, |vv_{i+1}|, \psi_{i+1})) \end{aligned} \quad (4.10)$$

By the definition of f we have to consider three cases in 4.10 for the contribution of a .

Case I: $\psi_i \geq \pi/2$ or $\psi_{i+1} \geq \pi/2$.

Without loss of generality assume that $\psi_i \geq \pi/2$. Then we know that $f(|vv_{i+1}|, |vv_i|, \psi_i) = 0$ and in the worst case $\pi_a(f(|vv_{i+2}|, |vv_{i+1}|, \psi_{i+1})) \leq |a|^2$. Thus we have that $\text{contribution}(a) \leq 5|a|^2$.

Case II: $\psi_i < \pi/2$ and $\psi_{i+1} < \pi/2$ and ($i = 1$ or ($i > 1$ and $\psi_{i-1} \geq \pi/2$)).

By definition of f we have:

$$\begin{aligned}\pi_a(f(|vv_{i+1}|, |vv_i|, \psi_i)) &= \pi_a(|vv_i|^2 + |vv_{i+1}|^2 \cos^2 \psi_i) = |a|^2 \cos^2 \psi_i \\ \pi_a(f(|vv_{i+2}|, |vv_{i+1}|, \psi_{i+1})) &= \pi_a((|vv_{i+1}|^2 + |vv_{i+2}|^2) \cos \psi_{i+1}) = |a|^2 \cos \psi_{i+1}\end{aligned}$$

Lemma 4.12 states that $\psi_i \geq (\pi - \psi_{i+1})/2$. We also know that ψ_i is at most π by definition. Thus we have:

$$\begin{aligned}\text{contribution}(a) &\leq (4 + \cos^2 \psi_i + \cos \psi_{i+1}) |a|^2 \\ &\leq \left(4 + \cos^2 \frac{\pi - \psi_{i+1}}{2} + \cos \psi_{i+1}\right) |a|^2 \\ &= \left(4 + \sin^2 \frac{\psi_{i+1}}{2} + \cos \psi_{i+1}\right) |a|^2\end{aligned}$$

Use $\cos 2x = \cos^2 x - \sin^2 x$.

$$\begin{aligned}&= \left(4 + \sin^2 \frac{\psi_{i+1}}{2} + \cos^2 \frac{\psi_{i+1}}{2} - \sin^2 \frac{\psi_{i+1}}{2}\right) |a|^2 \\ &= \left(4 + \cos^2 \frac{\psi_{i+1}}{2}\right) |a|^2 \\ &\leq 5|a|^2\end{aligned}$$

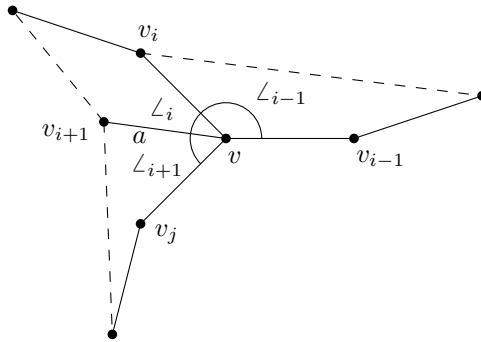


Figure 4.8: Illustration of Case III. If $\angle_{i-1} \geq \pi/2$ then $\angle_i + \angle_{i+1} < \pi$.

Case III: $\psi_i < \pi/2$ and $\psi_{i+1} < \pi/2$ and $i > 1$ and $\psi_{i-1} < \pi/2$.

We show that this leads to a contradiction. See Figure 4.8. We abbreviate $\angle\{v, v_i\}\{v, v_{i+1}\}$

with \angle_i . Note that $\angle_i = \pi - \psi_i$. Recall that the geometric T^3 -algorithm proceeds with the edge that makes the smallest angle with the current edge. The algorithm is first applied to edge $\{v, v_{i-1}\}$ and proceeds recursively with $\{v, v_i\}$. So we have that $\angle_{i-1} \leq \angle_j$ for some $j \geq i+1$. Since $\psi_{i-1} < \pi/2$ we have $\angle_{i-1} \geq \pi/2$ and thus also $\angle_j \geq \pi/2$. This leaves little room for the other angles: the shortest angle between $\{v, v_i\}$ and $\{v, v_j\}$ is at most π . This means that at least one of \angle_i and \angle_{i+1} is at most $\pi/2$ and thus at least one of ψ_i and ψ_{i+1} is greater than $\pi/2$, which contradicts the assumptions of case III.

In all cases the contribution of any edge $|a|$ to the tour is at most $5|a|^2$. The theorem follows by summing up the contribution of all edges. \square

4.4 Generalization to $\alpha \geq 2$

Theorem 4.15. *The geometric T^3 -algorithm is a $(3^{\alpha-1} + \sqrt{6}^\alpha/3)$ -approximation for TSP^α .*

Proof. If an edge a is used in a (≤ 2)-shortcut on one side and a (≤ 3)-shortcut on the other side then the total contribution of a to the tour is at most $(2^{\alpha-1} + 3^{\alpha-1})|a|^\alpha$ by Lemma 4.6. So we will focus our analysis again on the case that a is used in two 3-shortcuts. For $\alpha = 2$ we can express the weight of a 3-shortcut by Lemma 4.11 and rewrite the composite terms as in Theorem 4.14 inequality (4.8). Then we can apply Hölder's inequality for the case $\alpha > 2$.

$$\begin{aligned}
|s(a, b, c)|^\alpha &= (|s(a, b, c)|^2)^{\alpha/2} \\
&\leq (2|a|^2 + |b|^2 + 2|c|^2 + f(|a|, |b|, \psi_{ba}) + f(|c|, |b|, \psi_{bc}))^{\alpha/2} \\
&= ((2 + \pi_a(f(|a|, |b|, \psi_{ba})))|a|^2 + (1 + \pi_b(f(|a|, |b|, \psi_{ba})) + \pi_b(f(|c|, |b|, \psi_{bc})))|b|^2 \\
&\quad + (2 + \pi_c(f(|c|, |b|, \psi_{bc})))|c|^2)^{\alpha/2} \\
&= (\beta_a a^2 + \beta_b b^2 + \beta_c c^2)^{\alpha/2} \\
&\leq 3^{\alpha/2-1} (\beta_a^{\alpha/2} a^\alpha + \beta_b^{\alpha/2} b^\alpha + \beta_c^{\alpha/2} c^\alpha)
\end{aligned} \tag{4.11}$$

We've introduced the constants β to make the expression shorter. Hölder's inequality is applied to (4.11) with $p = \alpha/2$, $q = \frac{\alpha/2}{\alpha/2-1}$, $n = 3$, $x = (\beta_a a^2, \beta_b b^2, \beta_c c^2)$ and $y = (1, 1, 1)$. Note that the last inequality holds only by Hölder's inequality if $\alpha > 2$.

We will bound the contribution of an edge a that is used in two 3-shortcuts as in Theorem 4.14. Then there are three cases to consider. We've seen that the assumptions of Case III lead to a contradiction. So there are only two cases to consider.

Case I: $\psi_i \geq \pi/2$ or $\psi_{i+1} \geq \pi/2$.

$$\begin{aligned} \text{contribution}(a) &\leq 3^{\alpha/2-1} \left((2 + \cos \psi_i)^{\alpha/2} + (2 + \cos \psi_{i+1})^{\alpha/2} \right) |a|^\alpha \\ &\leq 3^{\alpha/2-1} (2^{\alpha/2} + 3^{\alpha/2}) |a|^\alpha \\ &= \left(3^{\alpha-1} + \frac{\sqrt{6}^\alpha}{3} \right) |a|^\alpha \end{aligned}$$

Case II: $\psi_i < \pi/2$ and $\psi_{i+1} < \pi/2$ and ($i = 1$ or ($i > 1$ and $\psi_{i-1} \geq \pi/2$)).

$$\text{contribution}(a) \leq 3^{\alpha/2-1} \left(\left(2 + \cos \psi_{i+1} \right)^{\alpha/2} + \left(2 + \sin^2 \frac{\psi_{i+1}}{2} \right)^{\alpha/2} \right) |a|^\alpha$$

To find the maximum of this term we compute the derivative of this term without the constant $3^{\alpha/2-1}$:

$$\begin{aligned} &-\frac{\alpha}{2} \left(2 + \cos \psi_{i+1} \right)^{\alpha/2-1} \sin \psi_{i+1} + \frac{\alpha}{2} \left(2 + \sin^2 \frac{\psi_{i+1}}{2} \right)^{\alpha/2-1} \cdot \sin \frac{\psi_{i+1}}{2} \cdot \cos \frac{\psi_{i+1}}{2} \\ &= \frac{\alpha}{2} \sin \psi_{i+1} \left(- \left(2 + \cos \psi_{i+1} \right)^{\alpha/2-1} + \frac{1}{2} \left(2 + \sin^2 \frac{\psi_{i+1}}{2} \right)^{\alpha/2-1} \right) \end{aligned}$$

So equating to zero yields one easy stationary point: $\sin(\psi_{i+1}) = 0$ and $0 \leq \psi_{i+1} < \frac{\pi}{2}$ implies $\psi_{i+1} = 0$. According to the graphs of Figure 4.9 the function is maximum at $\psi_{i+1} = 0$.

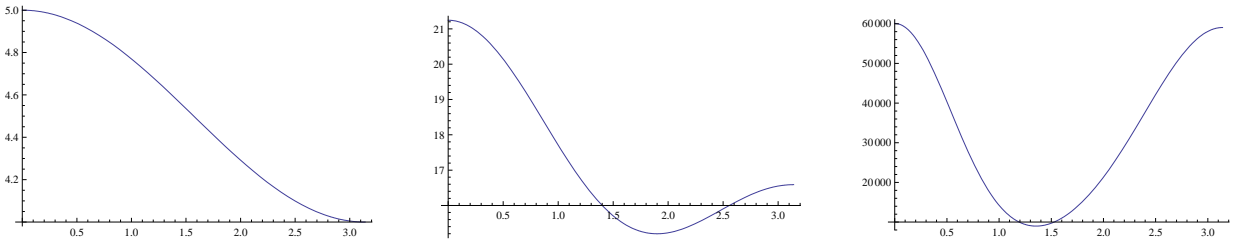


Figure 4.9: $(2 + \cos \psi_{i+1})^{\alpha/2} + \left(2 + \sin^2 \frac{\psi_{i+1}}{2} \right)^{\alpha/2}$ for $0 \leq \psi_{i+1} \leq \pi$ with $\alpha = 2, 5$ and 20 respectively.

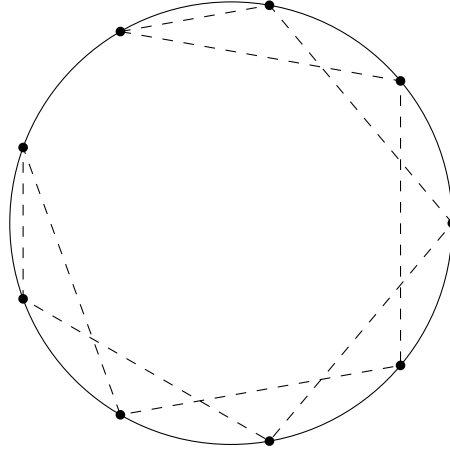


Figure 4.10: Example showing that the approximation ratio of T^3 -algorithm is at least 2^α .

The total contribution of a then becomes:

$$\begin{aligned} \text{contribution}(a) &\leq 3^{\alpha/2-1} \left(\left(2 + \cos \psi_{i+1} \right)^{\alpha/2} + \left(2 + \sin^2 \frac{\psi_{i+1}}{2} \right)^{\alpha/2} \right) |a|^\alpha \\ &\leq 3^{\alpha/2-1} \left(2^{\alpha/2} + 3^{\alpha/2} \right) |a|^\alpha \\ &= \left(3^{\alpha-1} + \frac{\sqrt{6}^\alpha}{3} \right) |a|^\alpha \end{aligned}$$

In all cases the contribution of any edge a to the tour is at most $(3^{\alpha-1} + \sqrt{6}^\alpha/3)|a|^\alpha$. The theorem follows for $\alpha > 2$ by summing up the contribution of all edges. The case $\alpha = 2$ corresponds to Theorem 4.14. \square

4.5 Lower bounds

Theorem 4.16. *The approximation ratio of all variants of the T^3 -algorithm is at least 2^α for TSP^α and for (d)SC.*

Proof. Consider a set of n points on a circle where consecutive points are unit distance apart. The MST of this point set is a simple path. If the T^3 -algorithm is applied to the first or last edge of the MST, the result will be a tour that contains almost only 2-shortcuts. See Figure 4.10. The tour will start from point 1 and continue to point 3 to point 5 etc. and finally to point n . From there it will continue to point $n-1$, $n-3$, $n-5$ etc. to point 2 and finally to point 1. If n approaches infinity then the angle between two edges of the MST tends to π . So the weight of a 2-shortcut tends to 2^α . The tour consist of $(n-2)$



Figure 4.11: The weight of this tour is $(3^{\alpha-1} + 1) \text{weight}(\text{MST})$

2-shortcuts and two 1-shortcuts. So the weight of the tour is $(n - 2)2^\alpha + 2$. The optimum TSP-tour just follows the points on the circle, hence the weight of the optimum is n . For the dSC problem the optimum also directs each antenna to the next antenna on the circle. The approximation ratio of the T^3 -algorithm is:

$$\lim_{n \rightarrow \infty} \frac{(n - 2)2^\alpha + 2}{n} = 2^\alpha.$$

□

We can give a better lower bound compared to the weight of the MST, showing that the analysis of Theorem 4.14 cannot be improved so much using only the same lower bound. We denote the resulting tour of the T^3 -algorithm with H .

Lemma 4.17. *If four points are on a straight line with unit distance between them and (any version of) the T^3 -algorithm is applied to the middle edge then*

$$\text{weight}(H) = (3^{\alpha-1} + 1) \text{weight}(\text{MST})$$

.

Proof. See Figure 4.11. The tour uses one 3-shortcut with weight equal to 3^α and the three edges of the MST with weight equal to 1. □

Some of our lower bound instances, such as the previous one, have edges of the MST in the final tour. If there is enough space we can improve the lower bound examples a little by adding points on the extensions of those edges and such that their distances follow a geometric series. See Figure 4.13 for the improvement of the previous instance. Before turning to the approximation ratio we first look at the weight of such series.

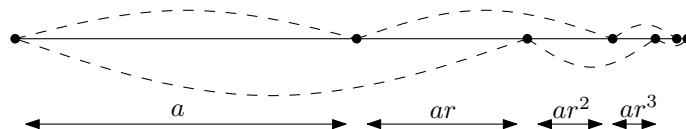


Figure 4.12: An example of a tour where n points are on a line and the inter distances of the points follow a geometric series with ratio r , $0 < r < 1$ and first distance is a .

Lemma 4.18. *Let the T^3 -algorithm be applied to the first edge of the MST of n points that are on a straight line and whose inter distances follow a geometric series with ratio r , $0 < r < 1$ and the first distance is a . Let H_n be the resulting tour, see Figure 4.12. Then we have that:*

$$\begin{aligned}\lim_{n \rightarrow \infty} \text{weight}(H_n) &= a^\alpha \left(1 + \frac{(1+r)^\alpha}{1-r^\alpha} \right) \\ \lim_{n \rightarrow \infty} \text{weight}(\text{MST}_n) &= \frac{a^\alpha}{1-r^\alpha}\end{aligned}$$

Proof. The tour consist of only 2-shortcuts and two 1-shortcuts at the ends. One 1-shortcut has weight a^α and the weight of the other 1-shortcut approaches zero when n approaches infinity. We can compute the weight of the 2-shortcuts by using geometric series.

$$\begin{aligned}\lim_{n \rightarrow \infty} \text{weight}(H_n) &= a^\alpha + (a + ar)^\alpha + (ar + ar^2)^\alpha + (ar^2 + ar^3)^\alpha + \dots \\ &= a^\alpha + (a(1+r))^\alpha \cdot (1 + r^\alpha + r^{2\alpha} + \dots) \\ &= a^\alpha + \frac{(a(1+r))^\alpha}{1-r^\alpha} \\ &= a^\alpha \left(1 + \frac{(1+r)^\alpha}{1-r^\alpha} \right)\end{aligned}$$

The weight of the MST is computed similarly.

$$\begin{aligned}\lim_{n \rightarrow \infty} \text{weight}(\text{MST}_n) &= a^\alpha + (ar)^\alpha + (ar^2)^\alpha + (ar^3)^\alpha + \dots \\ &= a^\alpha (1 + r^\alpha + r^{2\alpha} + r^{3\alpha} + \dots) \\ &= \frac{a^\alpha}{1-r^\alpha}\end{aligned}$$

□

We improve the example of Figure 4.11 by replacing the first and the last edge with a serie of points whose distances follow a geometric serie with ratio $1/2$. See Figure 4.13.

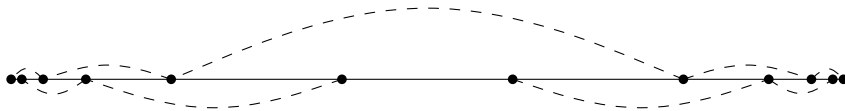


Figure 4.13: Improvement of the example of Figure 4.11 by taking two edges of the MST out of the tour.

Theorem 4.19. *If the T^3 -algorithm is applied to the middle edge of the MST of the points of Figure 4.13 then the ratio of the weight of the tour versus the weight of the MST is:*

$$\lim_{n \rightarrow \infty} \frac{\text{weight}(H_n)}{\text{weight}(\text{MST}_n)} = \frac{6^\alpha + 3^\alpha + 2^\alpha - 1}{3 \cdot 2^\alpha - 1}.$$

Proof. We prove something a bit more generic, by computing the ratio of the instance where the distance between the middle point and its two direct neighbors is a and the ratio of convergence is r . The general expression can be used to find a slightly better result for a given value of α . The resulting tour uses one 3-shortcut of length $(2a + 1)^\alpha$, the middle edge of the MST of length 1, and two series of 2-shortcuts. Using Lemma 4.18 the weight ratio is:

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{\text{weight}(H_n)}{\text{weight}(\text{MST}_n)} &= \frac{(2a + 1)^\alpha + 2 \frac{(a(r+1))^\alpha}{1-r^\alpha} + 1}{\frac{2a^\alpha}{1-r^\alpha} + 1} \\ &= \frac{2(a(r+1))^\alpha - ((2a + 1)^\alpha + 1)(r^\alpha - 1)}{2a^\alpha - r^\alpha + 1} \end{aligned} \quad (4.12)$$

Taking $a = 1$ and $r = \frac{1}{2}$, we get the following:

$$\begin{aligned} &= \frac{2(3/2)^\alpha - (1 + 3^\alpha)(1/2^\alpha - 1)}{3 - 1/2^\alpha} \\ &= \frac{(3/2)^\alpha + 3^\alpha - 1/2^\alpha + 1}{3 - 1/2^\alpha} \\ &= \frac{6^\alpha + 3^\alpha + 2^\alpha - 1}{3 \cdot 2^\alpha - 1} \end{aligned}$$

□

For $\alpha = 2$ we obtain $\text{weight}(H)/\text{weight}(\text{MST}) = 4 \frac{4}{11} \geq 4.36$. The reason that we've stated equation (4.12) in general form is that for a given value of α we can obtain a slightly better result. For $\alpha = 2$ we numerically obtained for $a = 1.073$ and $r = 0.518$ that $\text{weight}(H)/\text{weight}(\text{MST}) \geq 4.38$.

Chapter 5

Cycle-Insertion algorithm

In this chapter we'll look at another algorithm that is a 2-approximation for geometric TSP. The algorithm starts with a tour of length zero consisting of one vertex and incrementally adds one point at the time which is inserted somewhere in the current tour. The algorithm inserts a point by replacing an existent edge by two edges. They remove the edge that leads to the smallest increase in the length of the tour. There are multiple variations of this algorithm depending on:

- the order in which the points are being added and
- where to insert the next point in the current tour.

Rosenkrantz et al. [39] prove that two variants of the above algorithm are 2-approximations for TSP if the triangle inequality holds. The two algorithms vary in the order in which the points are being added:

- Nearest Insertion (NI): The next point is the point that is closest to the current tour. The distance from a point p to a tour T is defined as the minimum distance from p to any vertex in T .
- Cheapest Insertion (CI): The next point is *cheapest* to add to the current tour: it minimizes the increase in the length of the tour.

The fact that the NI method is a 2-approximation for geometric TSP is easiest to see.

Theorem 5.1 ([39]). *The NI method is a 2-approximation for geometric TSP.*

Proof. By Lemma 3.5 the MST is a lower bound for TSP. We denote the resulting tour of the NI-algorithm with T_{NI} and the optimal TSP-tour with T_{OPT} . We know that $\text{weight}(\text{MST}) \leq \text{weight}(T_{\text{OPT}})$ and we prove that $\text{weight}(T_{\text{NI}}) \leq 2 \cdot \text{weight}(\text{MST})$.

First of all note that there is a correspondence between the points that are added and the edges of the MST. At every iteration the point that is added to the tour is the point that

has the minimum distance to the tour. Prim's algorithm [19] to construct an MST selects the points in the same way: at every iteration a point that is closest to the current MST is added to the MST. The distance here is defined in the same way as for TSP: the distance from a point p to the MST is the minimum distance from p to any vertex of the MST. So the NI-algorithm could construct an MST at the same time it constructs a tour. When the algorithm picks a point p to add to the tour, the distance from p to the tour is defined by the distance between p and some point, q , in the tour. Prim's algorithm would now add the edge $e = \{p, q\}$ to the MST. Let $\text{cost}(T, p)$ be the increase in length of the tour T when the NI-algorithm adds p to it. If we can prove that $\text{cost}(T, p) \leq 2|e|$ then the theorem follows by summing up the cost of all iterations.

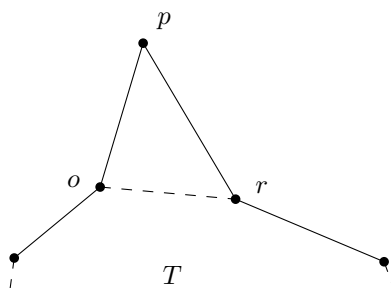


Figure 5.1: The NI-algorithm inserts p between o and r in tour T .

Assume that in one iteration we add the point p to the tour T between the points o and r , see Figure 5.1. Denote the increase in the length of the tour with $\text{cost}(T, o, p, r)$. The algorithm removes the edge $\{o, r\}$ from the tour and adds the edges $\{o, p\}$ and $\{p, r\}$ to the tour. So we have that:

$$\text{cost}(T, o, p, r) = |op| + |pr| - |or|. \quad (5.1)$$

By the triangle inequality we have that $|pr| \leq |op| + |or|$. Plugging this into (5.1) we get $\text{cost}(T, o, p, r) \leq 2|op|$. The point p is added in the cheapest way, so we take the minimum over all edges $\{o, r\}$ in T .

$$\text{cost}(T, p) = \min_{\{o,r\} \in T} \text{cost}(T, o, p, r) \leq 2|e|.$$

The theorem follows by summing up the costs of all iterations. \square

We have not been able to adapt the proof of Theorem 5.1 to power-weighted edges, but still we believe that the iterative method performs well in this case. We have at least been able to show that some variations of the NI-algorithm are constant-factor approximations.

5.1 NISE and NICE: two constant-factor approximations for TSP^α

We remark that in the proof of Theorem 5.1 it is not necessary to add the new point in the cheapest possible way. Instead, one can also add the new point before or after the closest point. Thus we get the following strategy:

1. As in the NI method, at each step the closest point p is added to the tour.
2. The new point p is added before or after the the point v_j in the current tour that is closest to it.

We consider two variants of this strategy. The two variants differ in the choice of inserting the point before or after v_j . Let v_{j-1} and v_{j+1} be the points that are neighbors of v_j in the tour. Define $\text{cost}(j, p)$ as the increase in the length of the tour if we insert point p between v_j and v_{j+1} .

1. The *Nearest Insertion Shortest Edge* method (NISE) inserts p between v_{j-1} and v_j if $|v_{j-1}v_j| \leq |v_jv_{j+1}|$ and between v_j and v_{j+1} otherwise.
2. The *Nearest Insertion Cheapest Edge* method (NICE) inserts p between v_{j-1} and v_j if $\text{cost}(j-1, p) \leq \text{cost}(j, p)$ and between v_j and v_{j+1} otherwise.

We have been able to prove that these two variations are constant-factor approximations by using a potential function. Let T_i be the tour after iteration i and MST_i the corresponding minimum spanning tree. The potential after iteration i of a node $v_j \in T_i$ is defined as:

$$\phi(v_j) = \kappa \cdot \min(|v_{j-1}v_j|, |v_jv_{j+1}|)^\alpha.$$

Here we take κ to be a constant such that $\kappa > 0$ and whose value we have yet to determine. The value of κ could depend on α . The total potential after iteration i is the sum of the potential of all points:

$$\phi(T_i) = \sum_{v \in T_i} \phi(v).$$

Using this potential function we show that the methods NISE and NICE are constant-factor approximations, where the upper bound on the approximation factor depends on the value of κ . We can then choose values of κ that minimize the upper bound on the approximation factor.

Lemma 5.2. *The NISE- and NICE-methods are c -approximations for TSP^α and (d)SC, where*

$$c = (1 + 2\kappa) \left(1 + (1 + \kappa) \left((1 + 2\kappa)^{1/(\alpha-1)} - (1 + \kappa)^{1/(\alpha-1)} \right)^{1-\alpha} \right).$$

Proof. We prove that the following invariant holds:

$$\text{weight}(T_i) + \phi(T_i) \leq c \cdot \text{weight}(\text{MST}_i).$$

Using that the MST is a lower bound for TSP^α and (d)SC by Lemmas 3.5 and 3.2 the theorem follows from this invariant.

The algorithm starts with one point and a tour of weight zero. So the invariant is immediately satisfied for $i = 0$.

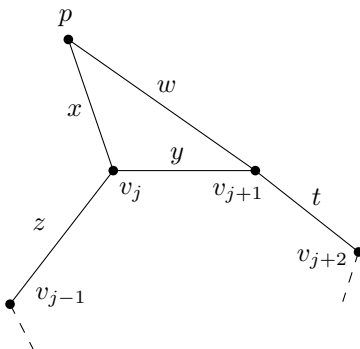


Figure 5.2: Point p is added to the tour between v_j and v_{j+1} because v_j is the closest point to p and $y \leq z$.

Now assume that the invariant holds after iteration i . We will show this implies that the invariant holds after iteration $i + 1$. We first consider only the NISE-method. Let p be the point that is added in iteration i and v_j be the point in T_i that is closest to it. Enumerate the other points in the tour as v_{j-1} , v_{j-2} and v_{j+1} , v_{j+2} etc, see Figure 5.2. Let $x = |pv_j|$, $y = |v_j v_{j+1}|$, $z = |v_{j-1} v_j|$, $t = |v_{j+1} v_{j+2}|$ and $w = |pv_{j+1}|$. Without loss of generality assume that $y \leq z$ so that the NISE-method inserts p between v_j and v_{j+1} .

Using the correspondence between the NI-strategy and the edges of the MST we know that $\text{weight}(\text{MST}_{i+1}) = \text{weight}(\text{MST}_i) + x^\alpha$. The potential function changes only at the points p , v_j and v_{j+1} . Let $\Delta f_i = f_{i+1} - f_i$ for a function f . Our proof obligation becomes:

$$\Delta \text{weight}(T_i) + \Delta \phi(T_i) \leq c \cdot \Delta \text{weight}(\text{MST}_i).$$

This is implied by:

$$\text{cost}(T_i, p) + \Delta \phi(p) + \Delta \phi(v_j) + \Delta \phi(v_{j+1}) \leq c \cdot x^\alpha. \quad (5.2)$$

We will look at the different components in (5.2) separately.

The algorithm adds the edges $\{p, v_j\}$ and $\{p, v_{j+1}\}$ to T_j and removes edge $\{v_j, v_{j+1}\}$. So we know that $\text{cost}(T_i, p) = x^\alpha + w^\alpha - y^\alpha$.

The potential of node p was 0 and becomes either κx^α or κw^α . We have $x \leq w$ because x is the shortest distance from p to any point in the tour. Thus we have $\Delta \phi(p) = \kappa x^\alpha$.

Since we have assumed that $y \leq z$ we only need to consider three cases for the potential of v_j .

$$\Delta\phi(v_j) = \begin{cases} \kappa(x^\alpha - y^\alpha) & \text{if } x \leq y \leq z, \\ \kappa(x^\alpha - y^\alpha) & \text{if } y \leq x \leq z, \\ \kappa(z^\alpha - y^\alpha) & \text{if } y \leq z \leq x. \end{cases}$$

In all cases we have $\Delta\phi(v_j) \leq \kappa(x^\alpha - y^\alpha)$.

For the change in potential of v_{j+1} we need to consider six cases.

$$\Delta\phi(v_{j+1}) = \begin{cases} \kappa(w^\alpha - t^\alpha) \leq 0 & \text{if } w \leq t \leq y, \\ \kappa(w^\alpha - y^\alpha) \leq 0 & \text{if } w \leq y \leq t, \\ \kappa(t^\alpha - t^\alpha) = 0 & \text{if } t \leq w \leq y, \\ \kappa(t^\alpha - t^\alpha) = 0 & \text{if } t \leq y \leq w, \\ \kappa(t^\alpha - y^\alpha) \leq w^\alpha - y^\alpha & \text{if } y \leq t \leq w, \\ \kappa(w^\alpha - y^\alpha) & \text{if } y \leq w \leq t. \end{cases}$$

In all cases we have $\Delta\phi(v_{j+1}) \leq \begin{cases} 0 & \text{if } w \leq y, \\ \kappa(w^\alpha - y^\alpha) & \text{if } w > y. \end{cases}$

Putting it all together we have:

$$\begin{aligned} \text{cost}(T_i, p) + \Delta\phi(T_i) &= \text{cost}(T_i, p) + \Delta\phi(p) + \Delta\phi(v_j) + \Delta\phi(v_{j+1}) \\ &\leq (x^\alpha + w^\alpha - y^\alpha) + \kappa \left(x^\alpha + (x^\alpha - y^\alpha) + \begin{cases} 0 & \text{if } w \leq y, \\ w^\alpha - y^\alpha & \text{if } w > y. \end{cases} \right) \\ &= (1 + 2\kappa)x^\alpha + w^\alpha - (1 + \kappa)y^\alpha + \begin{cases} 0 & \text{if } w \leq y, \\ \kappa(w^\alpha - y^\alpha) & \text{if } w > y. \end{cases} \end{aligned}$$

We consider the two cases separately.

Case I: $w \leq y$.

$$\begin{aligned} \text{cost}(T_i, p) + \Delta\phi &\leq (1 + 2\kappa)x^\alpha + w^\alpha - (1 + \kappa)y^\alpha \\ &\leq (1 + 2\kappa)x^\alpha - \kappa y^\alpha \\ &\leq (1 + 2\kappa)x^\alpha \end{aligned}$$

Case II: $w > y$. We use that $w \leq x + y$ by the triangle inequality.

$$\begin{aligned} \text{cost}(T_i, p) + \Delta\phi(T_i) &\leq (1 + 2\kappa)x^\alpha + (1 + \kappa)w^\alpha - (1 + 2\kappa)y^\alpha \\ &\leq (1 + 2\kappa)x^\alpha + (1 + \kappa)(x + y)^\alpha - (1 + 2\kappa)y^\alpha \end{aligned} \quad (5.3)$$

We show that in inequality (5.3) for every value of x there is a constant e , such that $y \leq e \cdot x$. Therefore we define $f_x(y) = (1 + 2\kappa)x^\alpha + (1 + \kappa)(x + y)^\alpha - (1 + 2\kappa)y^\alpha$. We can compute the extreme points of this function by taking the derivative:

$$f'_x(y) = (1 + \kappa)\alpha(x + y)^{\alpha-1} - (1 + 2\kappa)\alpha y^{\alpha-1}.$$

Equating $f'_x(y)$ to zero yields:

$$(1 + 2\kappa)y^{\alpha-1} = (1 + \kappa)(x + y)^{\alpha-1}. \quad (5.4)$$

Let $\alpha' = \frac{1}{\alpha-1}$. Then

$$\begin{aligned} (1 + 2\kappa)^{\alpha'} y &= (1 + \kappa)^{\alpha'} (x + y) \\ \left((1 + 2\kappa)^{\alpha'} - (1 + \kappa)^{\alpha'} \right) y &= (1 + \kappa)^{\alpha'} x \\ y &= \frac{(1 + \kappa)^{\alpha'} x}{(1 + 2\kappa)^{\alpha'} - (1 + \kappa)^{\alpha'}} \end{aligned}$$

Let $e = \frac{(1 + \kappa)^{\alpha'}}{(1 + 2\kappa)^{\alpha'} - (1 + \kappa)^{\alpha'}}$. Then

$$y = e \cdot x \quad (5.5)$$

The second-derivative test shows that $f_x(y)$ has a maximum at $y = e \cdot x$. Since it is the only extreme point, we have that $y \leq e \cdot x$ for all values of y and x . If we plug the value of (5.5) into (5.3) then it follows that $\text{cost}(T_i, p) + \Delta\phi(T_i) \leq g \cdot x^\alpha$ for a constant g . We can simplify the result somewhat by using equation (5.4):

$$\begin{aligned} (1 + \kappa)(x + y)^{\alpha-1} &= (1 + 2\kappa)y^{\alpha-1} \\ (1 + \kappa)(x + y)^\alpha &= (1 + 2\kappa)y^{\alpha-1}(x + y) \\ (1 + \kappa)(x + y)^\alpha &= (1 + 2\kappa)(y^\alpha + y^{\alpha-1}x) \\ (1 + \kappa)(x + y)^\alpha &\leq (1 + 2\kappa)(y^\alpha + e^{\alpha-1}x^\alpha) \end{aligned} \quad (5.6)$$

Plugging the value of (5.6) in (5.3) yields:

$$\begin{aligned} \text{cost}(T_i, p) + \Delta\phi(T_i) &\leq (1 + 2\kappa)x^\alpha + (1 + 2\kappa)e^{\alpha-1}x^\alpha \\ &= (1 + 2\kappa) \left(1 + \left(\frac{(1 + \kappa)^{\alpha'}}{(1 + 2\kappa)^{\alpha'} - (1 + \kappa)^{\alpha'}} \right)^{\alpha-1} \right) \\ &= (1 + 2\kappa) \left(1 + (1 + \kappa) \left((1 + 2\kappa)^{\alpha'} - (1 + \kappa)^{\alpha'} \right)^{1-\alpha} \right) x^\alpha \\ &= c \cdot x^\alpha \end{aligned}$$

The value of c is larger than the corresponding constant in the other case. So in both cases we have $\text{cost}(T_i, p) + \Delta\phi(T_i) \leq c \cdot x^\alpha$. This establishes the invariant for iteration $i + 1$.

We have shown that we can bound the cost of inserting one point in the tour and the cost of updating the potential function in the weight of one edge of the MST. So the NISE-algorithm is a c -approximation for all constant values of α .

Now consider the NICE-method. The proof is analogous to the proof of the NISE-method and uses the same potential function. The NICE-method also inserts the point p before or after v_j , but chooses the cheapest of the two. Now assume that $\text{cost}(j, p) \leq \text{cost}(j-1, p)$ so that p is inserted after v_j . If $y \leq z$ then the rest of the proof is the same as with the NISE-method. So assume that $y > z$. In this case the potential of point v_j was κz^α and now becomes either κx^α or stays κz^α . In both cases the difference of the potential of v_j is at most $\kappa(x^\alpha - z^\alpha)$. The differences in potentials of v_{j+1} and p are the same as with the NISE-method. Let $u = |v_{j-1}p|$. We know that $\text{cost}(j, p) \leq \text{cost}(j-1, p)$, which is the same as $x^\alpha + w^\alpha - y^\alpha \leq x^\alpha + u^\alpha - z^\alpha$. This implies $w^\alpha - y^\alpha \leq u^\alpha - z^\alpha$. Putting it all together we have:

$$\begin{aligned} \text{cost}(T_i, p) + \Delta\phi(T_i) &= \text{cost}(T_i, p) + \Delta\phi(p) + \Delta\phi(v_j) + \Delta\phi(v_{j+1}) \\ &\leq (x^\alpha + w^\alpha - y^\alpha) + \kappa \left(x^\alpha + (x^\alpha - z^\alpha) + \begin{cases} 0 & \text{if } w \leq y, \\ w^\alpha - y^\alpha & \text{if } w > y. \end{cases} \right) \\ &\leq (x^\alpha + u^\alpha - z^\alpha) + \kappa \left(x^\alpha + (x^\alpha - z^\alpha) + \begin{cases} 0 & \text{if } w \leq y, \\ u^\alpha - z^\alpha & \text{if } w > y. \end{cases} \right) \end{aligned}$$

The rest of the proof goes analogous with the roles of w and u , and y and z exchanged. \square

We've numerically obtained values for κ that minimize the upper bound on the approximation factor for some values of α .

Corollary 5.3. *The following table shows the upper bounds on the approximation factor of the NISE- and NICE-methods for some values of α .*

α	$\text{weight}(T_{\text{NISE}})/\text{weight}(\text{MST})$	κ
2	8	1/2
3	60.7	4/3
4	785.55	2.13
5	13762	2.86
6	296485	3.58

Table 5.1: Upper bounds on approximation factors for different values of α using the method of Lemma 5.2 and the value of the constant κ used in the potential function.

In the graph of Figure 5.3 we've shown the minimum value e such that $y \leq e \cdot x$ according to equation (5.5) for different values of α . The graph suggests that the value of e scales linearly with α . Therefore the obtained upper bound on the approximation factor of the two methods is proportional to α^α .

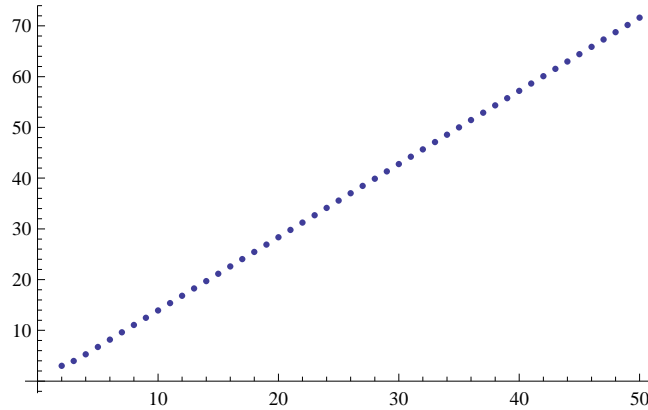


Figure 5.3: Plot that shows for a given value of α the value of e such that $y \leq e \cdot x$ in Lemma 5.2.

5.2 NISE with double update

We can change the NISE-method to further improve the approximation factor, but making the algorithm in spirit very similar to the T^3 -algorithm. First we observe that in the proof of the approximation factor of the NISE-method it is not strictly necessary to add the point p that has the closest distance to the tour. All that is required is that we add a point whose closest distance to the current tour makes an edge of the MST. So we can change the order in which we add points as follows:

1. Compute the MST.
2. Start with a tour of one point.
3. Take an edge $e = \{v_j, p\}$ of the MST such that v_j is in the tour and p is not in the tour.
4. Insert p in the tour.
5. Repeat

We can process the edges of the MST in any order, as long as one endpoint is in the tour when the other endpoint is inserted. Now we change the algorithm in two ways. Again let p be the point that is being inserted, let v_j be the point closest to p on the MST and suppose that $z = |v_j v_{j+1}| \leq |v_{j-1} v_j| = y$. Let $w = |p v_{j+1}|$ and $t = |v_{j+1} v_{j+2}|$, see Figure 5.4.

- After we've inserted a point p if there is another edge e of the MST adjacent to v_{j+1} and it holds that $y \leq t$ and $y \leq w$, then we insert the other endpoint of e immediately. We denote this endpoint with p' and let $x_2 = |e| = |p' v_{j+1}|$.

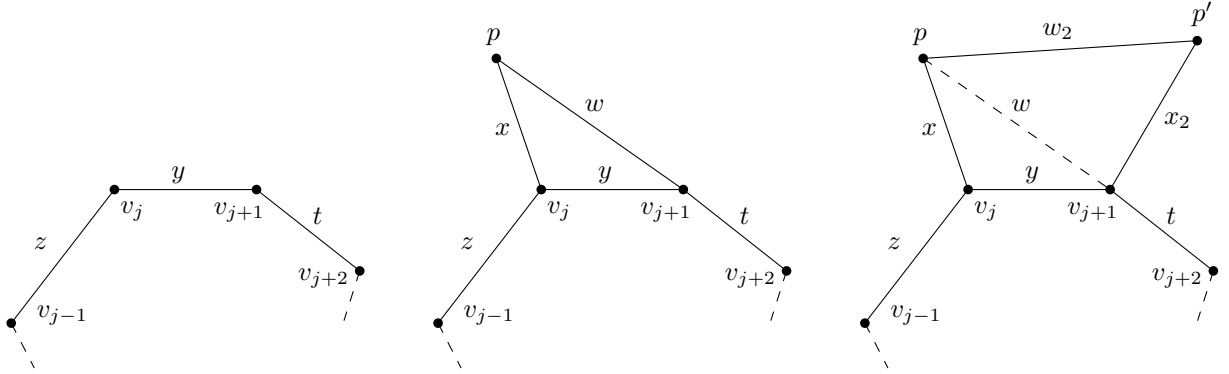


Figure 5.4: There are two possibilities for the dNISE-method to extend the current cycle.

- When we insert p' we remove the edge $\{p, v_{j+1}\}$ and we insert the edges $\{p, p'\}$ and $\{p', v_{j+1}\}$.

We refer to this method as dNISE. We show that dNISE is a $(2 \cdot 3^{\alpha-1})$ -approximation for $\alpha \geq 2$. This equals the approximation factor of the original T^3 -algorithm by Corollary 4.7 and is better than all upper bounds shown on the approximation factor of the NISE-method in Corollary 5.3.

Theorem 5.4. *The algorithm dNISE has an approximation factor of $3^{\alpha-1} + 2^\alpha - 1$ for $\alpha \leq 2$ and $2 \cdot 3^{\alpha-1}$ for $\alpha \geq 2$.*

Proof. The proof is partly analogous to the proof of Lemma 5.2. Again let T_i be the tour after iteration i . The potential of a node $v_j \in T_i$ is defined as:

$$\phi(v_j) = \begin{cases} \kappa \cdot \min(|v_{j-1}v_j|, |v_jv_{j+1}|)^\alpha & \text{if there is an edge } e = \{v_j, p\} \in T_i \text{ and } p \notin T_i, \\ 0 & \text{otherwise.} \end{cases}$$

Hence a vertex of T_i has zero potential if there is no other point of the MST directly adjacent to it that is not already in the tour. The value of κ will be different from the one in Lemma 5.2. Our invariant is:

$$\text{weight}(T_i) + \phi(T_i) \leq \text{weight}(\text{MST}_i).$$

To prove the theorem it suffices to show that after iteration i of the algorithm the increase in the weight of the tour plus the difference in the potential function can be bounded by the weight of the edge(s) of the MST that are added to MST_i . We know the following facts:

- $\text{cost}(T_i, p) = x^\alpha + w^\alpha - y^\alpha$ if the algorithm removes the edge $\{v_j, v_{j+1}\}$ and inserts the edges $\{p, v_j\}$ and $\{p, v_{j+1}\}$,

- $\Delta\phi(v_j) \leq \kappa(x^\alpha - y^\alpha)$, and
- $\Delta\phi(p) = \kappa \cdot x^\alpha$.

We consider four cases.

Case I: $w \leq y$.

In this case the algorithm inserts the edge $\{p, v_{j+1}\}$, just as in the NISE-method. So we know that $\Delta\phi(v_{j+1}) \leq \kappa(w^\alpha - y^\alpha) \leq 0$ and thus:

$$\Delta\text{weight}(T_i) + \Delta\phi \leq (1 + 2\kappa)x^\alpha.$$

Case II: $t \leq y \leq w$.

In this case the algorithm also inserts the edge $\{p, v_{j+1}\}$ as in the NISE-method. The potential of the point v_{j+1} doesn't change: both before and after iteration i it is $\kappa \cdot t^\alpha$. So we have that $\Delta\phi(v_{j+1}) = 0$.

$$\begin{aligned} \Delta\text{weight}(T_i) + \Delta\phi &\leq (x^\alpha + w^\alpha - y^\alpha) + \kappa(x^\alpha - y^\alpha) + \kappa \cdot x^\alpha \\ &= (1 + 2\kappa)x^\alpha + w^\alpha - (1 + \kappa)y^\alpha \\ &\leq (1 + 2\kappa)x^\alpha + (x + y)^\alpha - (1 + \kappa)y^\alpha \\ &\leq (1 + 2\kappa)x^\alpha + 2^{\alpha-1}(x^\alpha + y^\alpha) - (1 + \kappa)y^\alpha \\ &= (2^{\alpha-1} + 1 + 2\kappa)x^\alpha + (2^{\alpha-1} - 1 - \kappa)y^\alpha \end{aligned} \quad (5.7)$$

The fact that $(x + y)^\alpha \leq 2^{\alpha-1}(x^\alpha + y^\alpha)$ is proved in Lemma 4.6. From inequality (5.7) it follows that if we choose $\kappa \geq 2^{\alpha-1} - 1$ then $\Delta\text{weight}(T_i) + \Delta\phi(T_i) \leq (3 \cdot 2^{\alpha-1} - 1)x^\alpha$.

Case III: $y \leq w$ and $y \leq t$ there is no edge $e \in \text{MST}$ such that $e = \{v_j, p\}$ and $p \notin T_i$.

In this case the algorithm also inserts the edge $\{p, v_{j+1}\}$ as in the NISE-method. The potential of the point v_{j+1} becomes 0, so also the difference in potential is at most zero. The rest of case III is the same as case II.

Case IV: $y \leq w$ and $y \leq t$ and there is an edge $e \in \text{MST}$ such that $e = \{v_j, p\}$ and $p \notin T_i$.

In this case the algorithm deviates from the NISE-method and inserts the edges $\{v_j, p\}$, $\{p, p'\}$ and $\{v_{j+1}, p'\}$ as in Figure 5.4. Let $w_2 = |pp'|$. Then we can express the increase in the weight of the tour as $\Delta\text{weight}(T_i) = x^\alpha + w_2^\alpha + x_2^\alpha - y^\alpha$. The potential of p' becomes $\kappa \cdot x_2^\alpha$. The potential of v_{j+1} was $\kappa \cdot y^\alpha$ because $y \leq \min(w, t)$ and becomes $\kappa \min^\alpha(x_2, t) \leq \kappa \cdot x_2^\alpha$. Taking the terms together we get:

$$\begin{aligned} \Delta\text{weight}(T_i) + \Delta\phi(T_i) &\leq (x^\alpha + w_2^\alpha + x_2^\alpha - y^\alpha) + \kappa \cdot x^\alpha + \kappa \cdot x_2^\alpha + \kappa(x^\alpha - y^\alpha) + \kappa(x_2^\alpha - y^\alpha) \\ &= (1 + 2\kappa)(x^\alpha + x_2^\alpha) + w_2^\alpha - (1 + 2\kappa)y^\alpha \\ &\leq (1 + 2\kappa)(x^\alpha + x_2^\alpha) + (x + y + x_2)^\alpha - (1 + 2\kappa)y^\alpha \\ &\leq (1 + 2\kappa)(x^\alpha + x_2^\alpha) + 3^{\alpha-1}(x^\alpha + y^\alpha + x_2^\alpha) - (1 + 2\kappa)y^\alpha \\ &= (3^{\alpha-1} + 1 + 2\kappa)(x^\alpha + x_2^\alpha) + (3^{\alpha-1} - 1 - 2\kappa)y^\alpha \end{aligned} \quad (5.8)$$

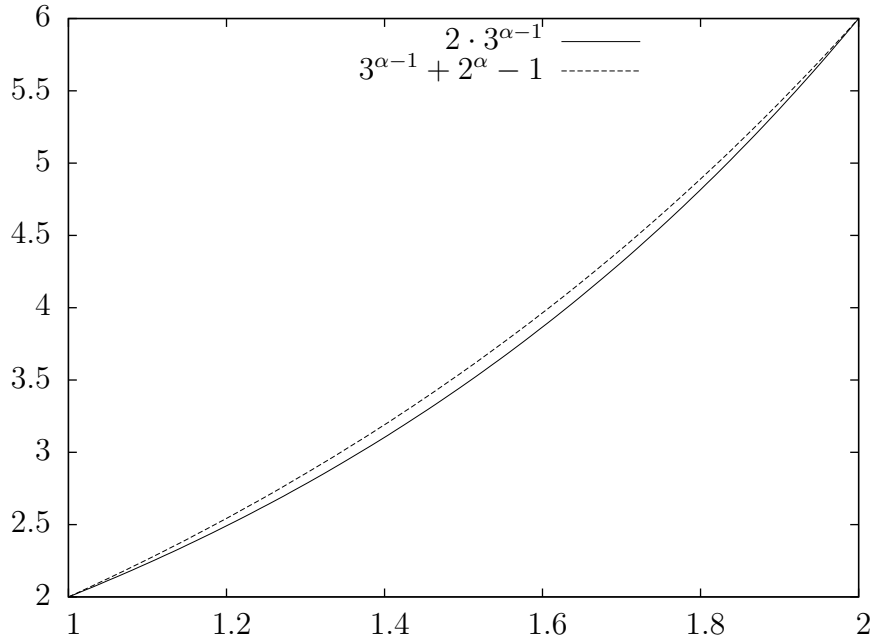


Figure 5.5: The two upper bounds on the the dNISE-method for $1 \leq \alpha \leq 2$.

The fact that $(x+y+x_2)^\alpha \leq 3^{\alpha-1}(x^\alpha+y^\alpha+x_2^\alpha)$ is proved in Lemma 4.6. From inequality 5.8 it follows that if we choose $\kappa \geq (3^{\alpha-1}-1)/2$ then $\Delta\text{weight}(T_i) + \Delta\phi(T_i) \leq 2 \cdot 3^{\alpha-1}(x^\alpha + x_2^\alpha)$.

The approximation factor of case IV is always more than the factors of cases I to III. To always bound the increase in the potential and the increase in the weight of the tour we should set κ to the maximum of $2^{\alpha-1} - 1$ and $(3^{\alpha-1} - 1)/2$. The maximum is $2^{\alpha-1} - 1$ for $\alpha \leq 2$ and $(3^{\alpha-1} - 1)/2$ for $\alpha \geq 2$. So we get an upper bound on the approximation factor of $3^{\alpha-1} + 2^\alpha - 1$ for $\alpha \leq 2$ and $2 \cdot 3^{\alpha-1}$ for $\alpha \geq 2$. \square

For $1 \leq \alpha \leq 2$ the approximation factor of dNISE, $3^{\alpha-1} + 2^\alpha - 1$, is slightly greater than $2 \cdot 3^{\alpha-1}$, the approximation factor for $\alpha \geq 2$ as Figure 5.5 shows.

5.3 Lower bounds

The lower bound example given in Theorem 4.16 for the T^3 -algorithm also applies for all our variants of the NI-algorithm.

Corollary 5.5. *The approximation ratio of the NI-, NISE- and NICE-algorithms is at least 2^α for TSP^α and for (d)SC.*

We've been able to find better lower bound instances using sophisticated constructions for the NI-, NISE- and NICE-algorithms where the weight of the tour divided by the weight of

the MST is more than 2^α . This shows that the upper bounds on the approximation factors cannot be improved so much using only the MST as a lower bound for the optimum. The lower bound examples are given in the Appendix. We summarize the results here.

In Theorem 5.6 we show that for the NISE-method there exists an instance on which it performs really bad.

Theorem 5.6. *For points in the plane there exists an instance such that*

$$\text{weight}(T_{\text{NISE}})/\text{weight}(\text{MST}) = \Omega(8^\alpha).$$

Furthermore there exists an instance for $\alpha = 2$ such that $\text{weight}(T_{\text{NISE}})/\text{weight}(\text{MST}) = 5$.

For the NICE-method we've even been able to find an instance on which the algorithm performs bad compared to the optimum.

Theorem 5.7. *The approximation factor of the NICE-algorithm is*

$$\Omega\left(\sqrt{8 + 3\sqrt{2}}^\alpha\right) = \Omega(3.49^\alpha).$$

The approximation factor of the NICE-algorithm is $\Omega(3.49^\alpha)$ whereas the approximation factor of the T^3 -algorithm is $O(3^\alpha)$ by Corollary 4.7. This means that the T^3 -algorithm eventually outperforms the NICE-algorithm for larger values of α .

Finally in Theorem 5.8 we give an instance on which the NI-method performs worse than 2^α .

Theorem 5.8. *There exists a set of points such that*

$$\text{weight}(T_{\text{NI}})/\text{weight}(\text{MST}) = \Omega\left(\sqrt{5 + 2\sqrt{2}}^\alpha\right) = \Omega(2.79^\alpha).$$

Chapter 6

NP-hardness

We will show that dSC is NP-hard by reducing planar 3SAT to it.

Definition 6.1. A *Boolean variable* x is a variable that can take the values true and false. A *literal* is an occurrence of a Boolean variable in a formula, either positive x , or negated $\neg x$. A *clause* is a conjunction of literals. A *3SAT* formula is a disjunction of clauses where each clause consist of exactly three literals. For example:

$$(x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_1 \wedge \neg x_3) \vee (x_2 \wedge \neg x_3 \wedge x_3).$$

A 3SAT formula is *satisfiable* if there is an assignment of values true and false to the variables such that all clauses evaluate to true. The 3SAT problem is: given any 3SAT formula ϕ , is it satisfiable?

Cook [18] has shown the 3SAT problem to be NP-complete.

Definition 6.2. Let ϕ be a 3SAT formula consisting of n Boolean variables x_1, \dots, x_n and m clauses c_1, \dots, c_m . The graph $G(\phi)$ is defined as follows:

- There is a vertex for every variable and for every clause.
- There is an edge $\{v_i, c_j\}$ if and only if variable v_i occurs in clause c_j .

The *planar* 3SAT problem is: given a 3SAT formula ϕ such that $G(\phi)$ is planar, is there an assignment of values true and false to the variables such that all clauses are satisfied?

Lichtenstein [32] shows that planar 3SAT is NP-complete.

Theorem 6.3. *The dSC problem is NP-hard for antennas with an angle less than $\pi/2$, that is, it is NP-hard to decide for a point set S whether there exists a range assignment for directed antennas with an angle less than $\pi/2$ of cost at most k such that the resulting communication graph is strongly connected.*

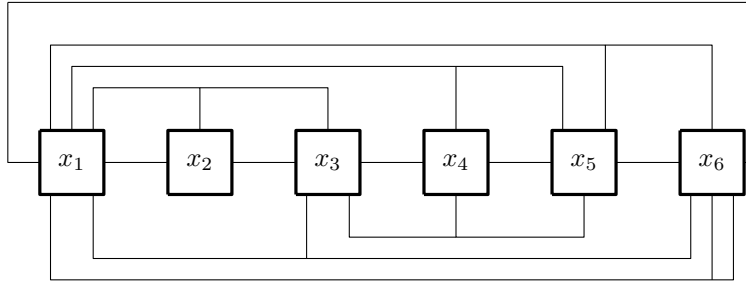


Figure 6.1: Embedding of a planar 3SAT formula with an additional cycle through the variables.

Proof. We show that the dSC problem is NP-hard by reducing planar 3SAT to it. For a given planar 3SAT formula ϕ we construct a point set S such that there exists a range assignment for directed antennas with the properties mentioned above of cost at most k if and only if ϕ is satisfiable. Knuth and Raghunathan [31] show that an instance of planar 3SAT can be embedded in the plane such that all variables x_1, \dots, x_n are drawn as boxes, positioned on a straight line and all clauses are drawn as three-legged combs above or below the variables, see Figure 6.1. The clauses are nested so that there are no intersections between the legs of the clauses. As in Figure 6.1 we draw an additional rectangle that contains all clauses drawn above the variables in its interior and intersects all the variable boxes.

We will use this drawing as a starting point and replace both the variables and the clauses with special gadgets. Since all clauses in the drawing are nested, we can make the drawing such that all edges are "sufficiently" far away from each other. The desired minimum distance between edges will depend on our construction. We place stations on the edges of the rectangle at unit distance from each other. We will also place stations on the clauses later on. Since the communication graph has to be strongly connected every station must be able to reach at least one other station. So any valid range assignment must assign a range to each station that is at least the minimum distance from that station to any other station. The minimum distance is not enough to reach other stations on clauses, since these are further away. Since we've assumed that the angle of all antennas is less than $\pi/2$ it is not possible to assign the minimum range to an antenna and direct the antenna such that it reaches two other stations on the rectangle at the same time. Hence a minimal range assignment directs each antenna so that it reaches exactly one of its two neighbors on the rectangle. The optimal range assignment directs all antennas on the rectangle clockwise or counter-clockwise to the next station to establish strong connectivity.

We replace the lower edge of the rectangle with *2-chains*, see Figure 6.2a. There is one 2-chain for every variable. Papadimitriou [35] uses 2-chains to prove that the Euclidean Traveling Salesman Problem is NP-hard. A 2-chain consists of two horizontal rows of stations, where there is for every station on the upper row a station on the lower row with the same x-coordinate and viceversa. The vertical distance between the two parallel rows

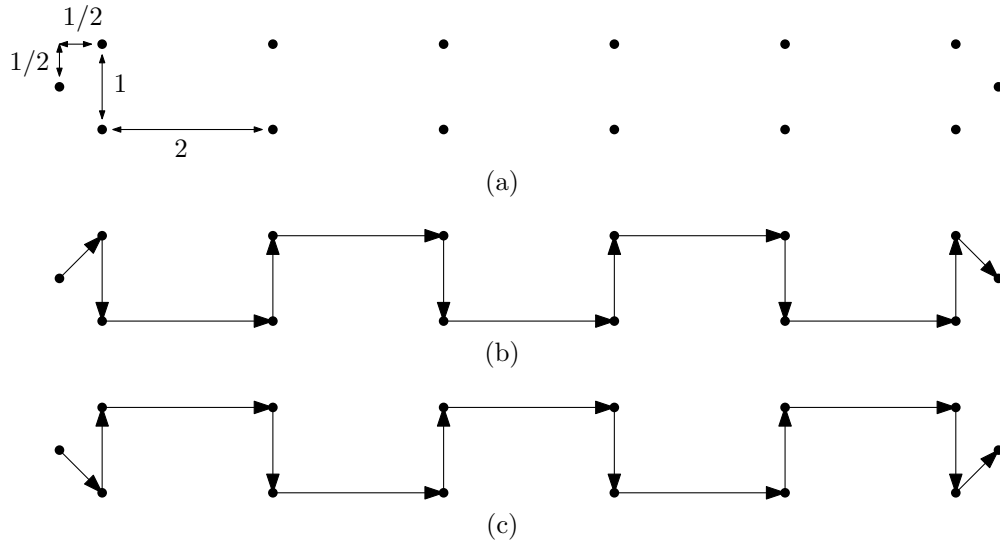


Figure 6.2: The 2-chain with two possible modes of pointing the antennas in an optimal way.

is 1 and the horizontal distances between stations on one row is 2. There is one additional station at each end of the 2-chain. Both the horizontal and vertical distance between such a station and the 2-chain is $1/2$.

The optimum range assignment still connects all stations on the rectangle in one big cycle. Therefore it must connect the stations in a 2-chain in either of two ways. The optimum can point the antennas in *mode 1* as in Figure 6.2b or in *mode 2* as in Figure 6.2c. With pointing we mean directing the antenna in such a way that the antenna can reach the other station. We will replace each variable with a 2-chain. Traversing the chain in mode 1 will respond to setting the variable to true and traversing the chain in mode 2 will respond to setting the variable to false.

We replace each clause with the configuration of Figure 6.3. In this gadget we replace each of the four line segments of the three-legged comb with two parallel rows of stations, but no stations are placed between the two rows of another line segment. The distance between two parallel rows is 4. The configuration is placed at a distance a from the 2-chains. A suitable constant for a is 4. Each leg of the clause has a narrower end with a width of 2 which is lined up with the 2-chain of the variable to which the leg corresponds. This can be done in two ways: the end can be positioned to the left or to the right. We determine this position later on.

Let us first look at how an optimal solution connects all the stations on a clause to the main cycle, independent of formula ϕ . An optimal solution needs to bridge the distance between the 2-chains and a clause only twice. This can be done by cutting the cycle in one 2-chain and pointing one of the antennas in the 2-chain to the nearest station in the clause. Then traverse the whole circumference of the clause and return to the 2-chain. We assure

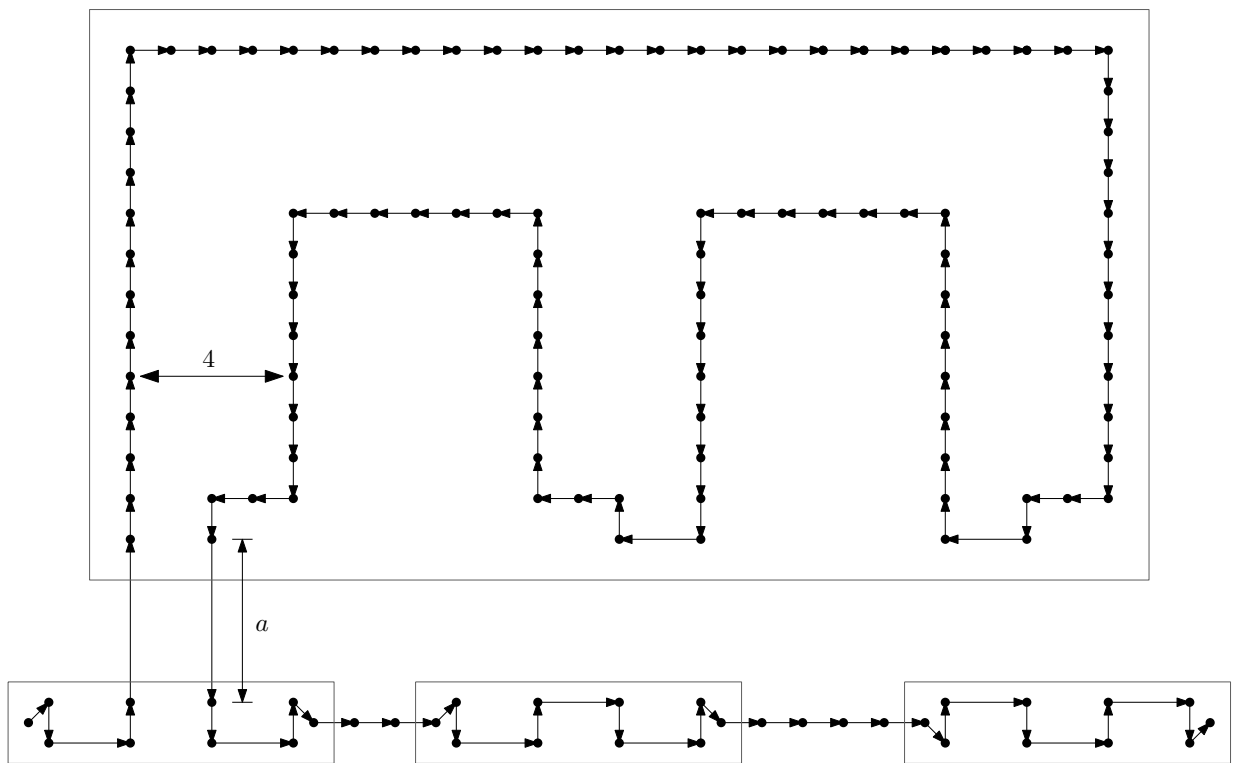


Figure 6.3: The construction for a single clause drawn together with three 2-chains and an optimal assignment of antennas.

that the distance from a clause to other clauses is more than a . So any other solution would require bridging the a distance more than twice.

A clause can be connected to the main cycle from any of the three 2-chains. It is cheapest, however, if the used 2-chain is in the right mode. If all three 2-chains are in the wrong mode, then a minimal solution is forced to use a more expensive range assignment to establish strong connectivity in the communication graph. We can make sure that the clause can be connected from a 2-chain in the right mode if ϕ is satisfiable by changing the positions of the narrower ends of the legs of the clause to the left or to the right. Recall that the three legs correspond to the three literals in the clause. A variable can occur positive or negated in the clause. If the occurrence of the variable of a leg is positive, we change the position of the end of the leg so that the clause can be reached cheaply if the 2-chain is used in mode 1. Subsequently if the variable occurs negated, we change the position so that the clause can be reached cheaply if the 2-chain is used in mode 2.

We show that for a given planar 3SAT formula ϕ the construction with the gadgets described above admits a range assignment for directed antennas such that the resulting communication graph is strongly connected of cost at most k if and only if ϕ is satisfiable.

If. If the 3SAT formula ϕ is satisfiable then there exists a satisfying truth assignment to the variables and we can set the 2-chains to the corresponding modes. Let l denote the total cost when we make all stations on the rectangle strongly connected by creating one big cycle and let l_i be the total cost when we make clause i strongly connected by creating one big cycle along the circumference of the clause. The cost of the optimal range assignment if ϕ is satisfiable is $k = l + \sum_{1 \leq i \leq m} l_i + m(2a - 2)$.

Only if. If the cost of the optimal range assignment is at most k then the optimal range assignment connects every clause optimally from one of the 2-chains. This means that the 2-chains are used in the right mode, and this implies a truth assignment to the variables of ϕ that makes ϕ true.

Our whole construction is polynomial in size. Thus the dSC problem is NP-hard for antennas with an angle less than $\pi/2$. \square

Chapter 7

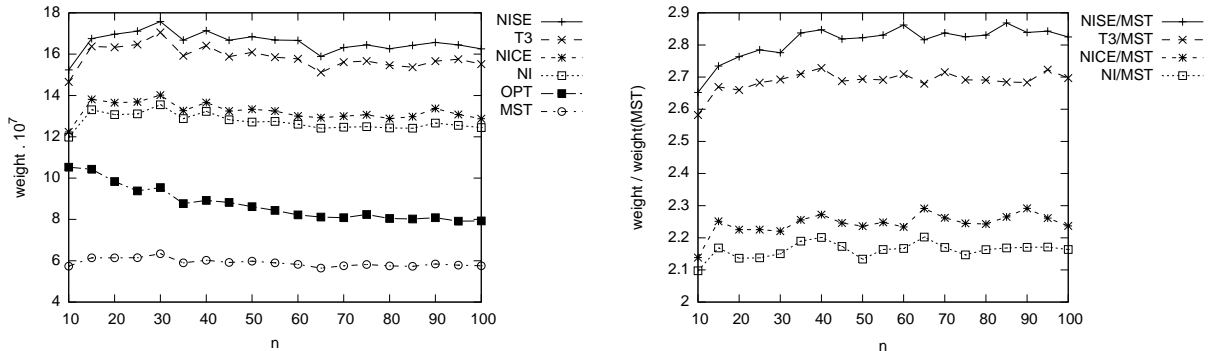
Experiments

We've implemented the geometric T^3 -, NI-, NISE- and NICE-algorithms. We've tested the performance both on random instances and on instances from the TSPLIB library [38]. We've not implemented the dNISE-method, because it is a late addition to this thesis.

7.1 Random tests

We've experimented by varying the number of nodes n and by varying the distance-power gradient α . For each experiment with fixed parameters n and α we've generated 50 instances by choosing n points uniformly at random from a regular grid of size 10 000 by 10 000. We've taken the average of the weights of the solutions among the 50 instances. Figure 7.1a shows the average weight of the tours after 50 runs for $\alpha = 2$ and for $10 \leq n \leq 100$ in increments of 5. We've also computed the optimal tour for TSP² using the Concorde TSP Solver [8]. We've not been able to compute the exact solution to the dSC problem. Therefore we've computed the ratio of the weight of every tour divided by the weight of the MST, again averaging over the 50 instances. The results are shown in Figure 7.1b. Lemma 3.1 says that $\text{weight}(r_{\text{OPT}}) \geq \text{weight}(\text{MST})$ and so $\text{weight}(r)/\text{weight}(r_{\text{OPT}}) \leq \text{weight}(r)/\text{weight}(\text{MST})$ for every range assignment r . So the ratio of the weight of a tour divided by the weight of the MST is an upper bound for the performance.

Figure 7.1a indicates that weights of the tours and the MST don't change much when we have more cities (inside the same square). Figure 7.1b shows that the ratios of the tours are as expected less than what we've proved. What is interesting is the ordering of the ratios of the different algorithms. Although we have a better upper bound on the approximation factor for the T^3 -algorithm than for the NICE-algorithm, the performance of T^3 is consistently worse. The NICE-algorithm is only slightly worse than the NI-algorithm, so this justifies the adaptation of the NI-algorithm for the sake of proving the approximation factor. The NISE-algorithm gives the worst results, so this is not a good adaptation of the NI-algorithm.



(a) Weights of the tours of the different algorithms. (b) Ratio of weights of the tours over the weight of the MST.

Figure 7.1: Results of the first experiment using $\alpha = 2$.

In the second experiment we've varied the parameter α . The results of the first experiment indicates that the weights of the tours don't depend much on the number of cities, so we've taken n to be 30. We've experimented with values for α between 1 and 6 in increments of 0.5. In this experiment we haven't been able to compute the optimal tour with the Concorde TSP-solver. The reason is that the Concorde program assumes that edge weights can be represented by 16-bit signed integers. The maximum edge weight in our square is $\sqrt{2} \cdot 10.000^\alpha \leq \sqrt{2} \cdot 10.000^6 = 8 \cdot 10^{12} \leq 2^{43}$. Hence we would need at least 43 bits if we round the non-integer weights for the case that α is not an even integer. If we had more bits available (say, 64) we could multiply all weights with an integer before rounding to compute the optimum tour more reliably. Alternatively we could have chosen a smaller grid size, but using 16-bit signed integers and $\alpha \leq 6$ only allows for a regular grid of size 16 by 16 for the case that α is an even integer. Therefore we decide to do without optimal solutions and instead compare to the weight of the MST. Figure 7.2 shows the ratios of the weights of the tour compared to the weight of the MST in the second experiment.

The experiment indicates that the approximation ratios of the algorithms indeed grow exponentially, but for random instances the base of the exponent is a lot less than what we've been able to prove. We have the same ordering in performance as in the previous experiment, but the results suggest that the difference in the ratios of the different algorithms becomes larger if α is larger. Hence the experiments suggest that the performance of the NI- and NICE-algorithms is in fact exponentially better than the factor of the T³-algorithm, contrary to our best upper bounds on the approximation factors for these algorithms.

7.2 TSPLIB

We've also experimented with real-life instances from the TSPLIB library. From the TSPLIB library we've taken the instances from the symmetric TSP problem where the

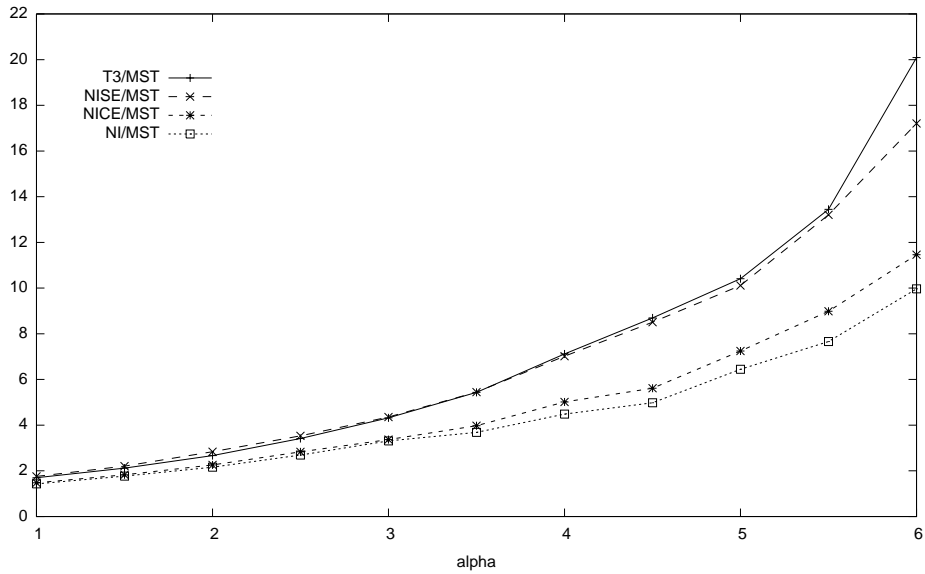


Figure 7.2: The weight ratios of the different algorithms for some values of α .

Euclidean measure is used. Each instance is given as a list of points in the plane. One instance (the Lin/Kernighan Original 318-city problem) requires a certain edge to be in the solution tour. We've ignored this requirement. We've experimented with values for α between 1 and 10 in increments of 1. For each value of α we've run all algorithms on all instances. To normalize the results we've divided the weight of a tour by the weight of the MST. Figure 7.3 shows the results for $\alpha = 2$. We see that also for instances of the TSPLIB library the performances of the different algorithms are ordered in the same way as when run on random instances. Bearing some exceptions, the NI-algorithm does best, followed by the NICE-method, next is the geometric T^3 -algorithm and worst is the performance of the NISE-method. For the other values of α we've experimented with, we see the same pattern. The results are shown in Figure A.7 in the appendix.

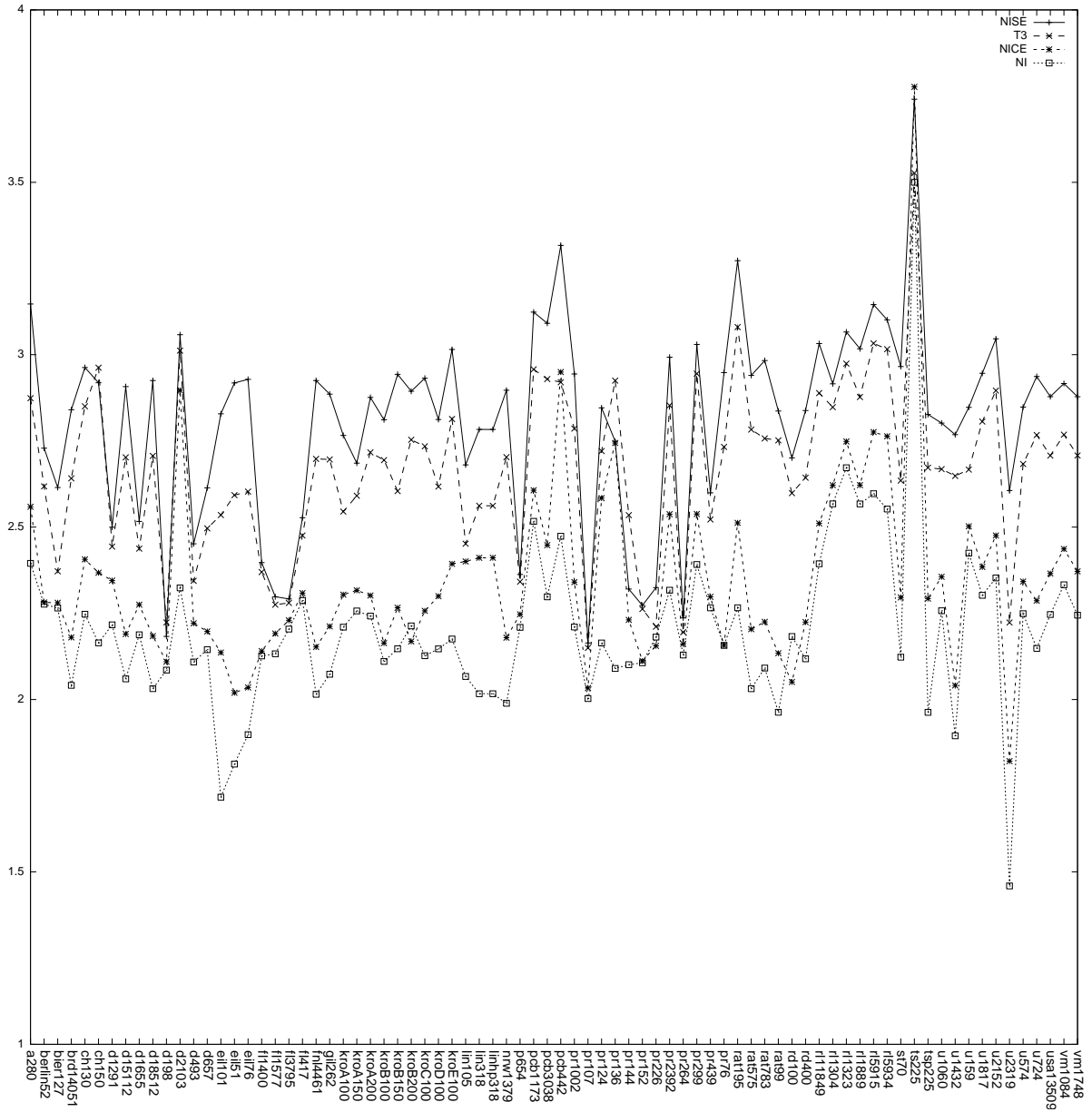


Figure 7.3: The weight ratios of the different algorithms on the instances from TSPLIB for $\alpha = 2$.

Chapter 8

Conclusion

We've shown that the dSC problem is NP-hard for antennas with an angle smaller than $\pi/2$. Therefore we've studied approximation algorithms. We've shown that two approximation algorithms for geometric TSP can be adapted to work for TSP^α and that this also implies an approximation algorithm for the (d)SC problem. We've shown that the approximation factor of the T^3 -algorithm of Andreae and Bandelt [7] for these problems is $2 \cdot 3^{\alpha-1}$ and we've been able to improve this to $(3^{\alpha-1} + \sqrt{6}^\alpha/3)$ for $\alpha \geq 2$ by adapting the algorithm. For TSP^α the algorithm of Bender and Chekuri implies an approximation factor of $2^{\alpha+1}$. Hence our algorithm improves the approximation factor for $2 \leq \alpha \leq 3.419$ for TSP^α . The result of Bender and Chekuri doesn't carry over easily to dSC. A lower bound example shows that the approximation ratio of both the T^3 -algorithm and the NI-algorithm and its variants is $\Omega(2^\alpha)$. For the T^3 -algorithm and its variants we've shown that the weight of the tour versus the weight of the MST for $\alpha = 2$ can be more than 4.38. This brings the upper bound of 5 for $\alpha = 2$ close to the lower bound.

For the NISE- and NICE-methods we've been able to show that both methods are constant-factor approximation algorithms for constant α . For a given value of α we can numerically obtain an approximation factor, but we haven't been able to express the approximation factor as a function of α . The constant is 8 for $\alpha = 2$ and seems to grow hyperexponentially with α . We've shown that for the NISE-, NICE- and NI-algorithms there are worse lower bound examples compared to the MST than the $\Omega(2^\alpha)$ lower bound. For the NISE-method the lower bound example shows that the ratio of the weight of the tour versus the weight of the MST grows at least as $\Omega(8^\alpha)$, which shows that the approximation factor of the NISE-method cannot be improved much using only the MST as a lower bound for the optimum. For the NICE-method we've shown that the approximation factor is at least $\Omega(3.49^\alpha)$, which means that the T^3 -algorithm eventually outperforms the NICE-method.

We've experimented both with random data and with real-life instances from the TSPLIB library. Our experiments suggest that for practical (small) values of α there is a clear ranking among the algorithms: NI does best, NICE does slightly worse than NI, T^3 does worse than NICE and NISE does a little worse than T^3 . So our theory doesn't do justice to the NI-algorithm and its adaptation, NICE. The experiments suggest that the approxi-

mation factor of NICE could in fact be better for small values of α —even better than the approximation factor of the T^3 -algorithm.

Concerning future research we think that our results might be improved upon in two ways. As mentioned above we expect that the approximation factor of the NI-algorithm is closer to the lower bound of 2^α and better than the T^3 -algorithm, at least for small values of α . The only geometric information that we use to prove that the NISE-method is a constant-factor approximation is the triangle inequality. A tighter analysis should make use of angular information just like our proof of the T^3 -algorithm.

In our proof of the approximation ratio of the T^3 -algorithm we study the generation of two adjacent 3-shortcuts. The approximation ratio of the T^3 -algorithm for TSP^α might be improved by taking more consecutive steps of the algorithm into account. The number of 3-shortcuts that can occur around one vertex is limited by the degree of the vertex. The degree of a vertex of the MST is at most six by Lemma A.1, but the more 3-shortcuts occur around one vertex, the less space there is for each 3-shortcut and the cheaper the shortcuts get. On the other hand one edge of the MST can easily have a contribution of 5 for $\alpha = 2$ if it is used in one 3-shortcut and one 2-shortcut. Due to the small node degree, however, the algorithm cannot continue in this way: the next shortcut that the algorithm generates is a (≤ 2)-shortcut, so the contribution of the expensive edge is compensated by the contribution of the other edge.

In our algorithms we haven't been able to exploit that directed antennas can reach more than one other station if there are multiple stations in the antenna's beam. Very recently Caragiannis et al. [12] have investigated a problem that differs from dSC only in that every station must be assigned the same range to achieve strong connectivity. They give a $2 \sin(\pi - \phi/2)$ -approximation for $\pi \leq \phi \leq 8\pi/5$, where ϕ denotes the angle of the antenna's beam. For big angles of ϕ they do exploit that one antenna can reach more than one other antenna. Their method probably also works for the dSC-problem. Another point of interest is that they use a hexagonal grid to show NP-hardness of their problem for antennas with an angle smaller than $2\pi/3$ as opposed to $\pi/2$.

Bibliography

- [1] E. Althaus, G. Călinescu, I. I. Mandoiu, S. Prasad, N. Tchervenski, and A. Zelikovsky. Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. *Wireless Networks*, 12(3):287–299, 2006.
- [2] C. Ambühl. An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In L. Caires, G. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 3580 of *LNCS*, pages 1139–1150. Springer-Verlag, 2005.
- [3] C. Ambühl, A. Clementi, M. Di Ianni, N. Lev-Tov, A. Monti, D. Peleg, G. Rossi, and R. Silvestri. Efficient algorithms for low-energy bounded-hop broadcast in ad-hoc wireless networks. In V. Diekert and M. Habib, editors, *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *LNCS*, pages 418–427. Springer-Verlag, 2004.
- [4] C. Ambühl, A. Clementi, M. Di Ianni, G. Rossi, A. Monti, and R. Silvestri. The range assignment problem in non-homogeneous static ad-hoc networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 224–232. IEEE Computer Society, 2004.
- [5] C. Ambühl, A. Clementi, P. Penna, G. Rossi, and R. Silvestri. On the approximability of the range assignment problem on radio networks in presence of selfish agents. *Theoretical Computer Science*, 343(1-2):27–41, 2005.
- [6] T. Andreae. On the traveling salesman problem restricted to inputs satisfying a relaxed triangle inequality. *Networks*, 38(2):59–67, 2001.
- [7] T. Andreae and H. Bandelt. Performance guarantees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics*, 8(1):1–16, 1995.
- [8] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde: A code for solving traveling salesman problems, 2003. <http://www.tsp.gatech.edu/concorde/>.
- [9] M. A. Bender and C. Chekuri. Performance guarantees for the TSP with a parameterized triangle inequality. *Information Processing Letters*, 73(1-2):17–21, 2000.

- [10] G. Călinescu, S. Kapoor, A. Olhevsky, and A. Zelikovsky. Network lifetime and power assignment in ad hoc wireless networks. In G. Di Battista and U. Zwick, editors, *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, volume 2832 of *LNCS*, pages 114–126. Springer-Verlag, 2003.
- [11] I. Caragiannis, M. Flammini, and L. Moscardelli. An exponential improvement on the mst heuristic for minimum energy broadcasting in ad hoc wireless networks. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 447–458. Springer-Verlag, 2007.
- [12] I. Caragiannis, C. Kaklamanis, E. Kranakis, D. Krizanc, and A. Wiese. Communication in wireless networks with directional antennas. In F. M. auf der Heide and N. Shavit, editors, *Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures (SPA)*, pages 344–351, 2008.
- [13] P. Carmi, M. Katz, and J. Mitchell. The minimum-area spanning tree problem. *Computational Geometry: Theory and Applications*, 35(3):218–225, 2006.
- [14] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca. On the complexity of computing minimum energy consumption broadcast subgraphs. In A. Ferreira and H. Reichel, editors, *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2010 of *LNCS*, pages 121–131. Springer-Verlag, 2001.
- [15] A. Clementi, M. Di Ianni, and R. Silvestri. The minimum broadcast range assignment problem on linear multi-hop wireless networks. *Theoretical Computer Science*, 299(1–3):751–761, 2003.
- [16] A. Clementi, A. Ferreira, P. Penna, S. Perennes, and R. Silvestri. The minimum range assignment problem on linear radio networks. *Algorithmica*, 35(2):95–110, 2003.
- [17] A. Clementi, P. Penna, and R. Silvestri. On the power assignment problem in radio networks. *Mobile Networks and Applications*, 9(2):125–140, 2004.
- [18] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.
- [19] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 2nd edition, 2001.
- [20] G. Das, S. Das, and S. Nandy. Range assignment for energy efficient broadcasting in linear radio networks. *Theoretical Computer Science*, 352(1–3):332–341, 2006.

- [21] G. Das, S. Ghosh, and S. Nandy. An efficient heuristic algorithm for 2D h -hops range assignment problem. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1051–1055, 2004.
- [22] G. Das, S. Ghosh, and S. Nandy. Improved algorithm for minimum cost range assignment problem for linear radio networks. *International Journal of Foundations of Computer Science*, 18(3):619–635, 2007.
- [23] M. Di Ianni and G. Rossi. Minimum energy range assignment in heterogeneous ad-hoc networks. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, pages 121–126. IEEE Computer Society, 2007.
- [24] M. Flammini, R. Klasing, A. Navarra, and S. Perennes. Improved approximation results for the minimum energy broadcasting problem. *Algorithmica*, 49(4):318–336, 2007.
- [25] B. Fuchs. On the hardness of range assignment problems. *Networks*, 52(2), 2008. To appear. Online available at: <http://dx.doi.org/10.1002/net.20227>.
- [26] S. Funke and S. Laue. Bounded-hop energy-efficient broadcast in low-dimensional metrics via coresets. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *LNCS*, pages 272–283. Springer-Verlag, 2007.
- [27] E. Kantor and D. Peleg. Approximate hierarchical facility location and applications to the shallow steiner tree and range assignment problems. In T. Calamoneri, I. Finocchi, and G. F. Italiano, editors, *Proceedings of the 6th Italian Conference on Algorithms and Complexity (CIAC)*, volume 4235, pages 211–222. Springer-Verlag, 2006.
- [28] S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21(2):434–450, 1996.
- [29] L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 243(1–2):289–305, 2000.
- [30] R. Klasing, A. Navarra, and A. Papadopoulos. Adaptive broadcast consumption (ABC), a new heuristic and new bounds for the minimum energy broadcast routing problem. In N. Mitrou, K. P. Kontovasilis, G. N. Rouskas, I. Iliadis, and L. F. Merakos, editors, *Proceedings of the 3th International IFIP-TC6 Networking Conference*, volume 3042 of *LNCS*, pages 866–877. Springer-Verlag, 2004.
- [31] D. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.
- [32] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329, 1982.

- [33] A. Navarra. Tighter bounds for the minimum energy broadcasting problem. In *Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WIOPT)*, pages 313–322. IEEE Computer Society, 2005.
- [34] A. Navarra. 3-dimensional minimum energy broadcasting problem. *Ad Hoc Networks*, 6(5):734–743, 2008.
- [35] C. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.
- [36] J. Park and S. Sahni. Power assignment for symmetric communication in wireless sensor networks. In *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC)*, pages 591–596, 2006.
- [37] J. Park and S. Sahni. Power assignment for symmetric communication in wireless sensor networks. *International Journal on Distributed Sensor Networks*, 2008. to appear.
- [38] G. Reinelt. TSPLIB—a traveling salesman library. *ORSA Journal on Computing*, 3(4):376–384, 1991.
- [39] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.
- [40] M. Sekanina. On an ordering of the set of vertices of a connected graph. *Publications of the Faculty of Science, University of Brno*, 412:137–142, 1960.
- [41] P. J. Wan, G. Călinescu, X. Y. Li, and O. Frieder. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless Networks*, 8(6):607–617, 2002.
- [42] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 585–594, 2000.

Appendix

A.1 Additional proofs

Lemma 4.6. Let e be a k -shortcut that uses edges e_1, \dots, e_k . Then the following inequality holds:

$$|e|^\alpha \leq k^{\alpha-1} \sum_{1 \leq i \leq k} |e_i|^\alpha. \quad (\text{A.1})$$

Proof. The proof uses the same arguments as in the proof of Lemma 4.3. By repeating the triangle inequality we have:

$$|e|^\alpha \leq \left(\sum_{1 \leq i \leq k} |e_i| \right)^\alpha. \quad (\text{A.2})$$

And by Hölder's inequality with $p = \alpha$, $q = \alpha/(\alpha - 1)$, $x = (|e_1|, \dots, |e_k|)$ and $y_i = 1$ for $0 \leq i \leq k$:

$$\sum_{1 \leq i \leq k} |e_i| \leq \left(\sum_{1 \leq i \leq k} |e_i|^\alpha \right)^{\frac{1}{\alpha}} k^{\frac{\alpha-1}{\alpha}}.$$

Because $\alpha \geq 1$ this implies:

$$\left(\sum_{1 \leq i \leq k} |e_i| \right)^\alpha \leq k^{\alpha-1} \sum_{1 \leq i \leq k} |e_i|^\alpha. \quad (\text{A.3})$$

Inequalities (A.2) and (A.3) together imply (A.1). \square

Lemma A.1. Let P be a set of points in the plane and let $G = (P, P^2; w)$ be the complete weighted graph with weight function $w(u, v)$ given by $w(u, v) = |uv|^\alpha$. Two edges of the MST of G that share an endpoint must make an angle of at least $\pi/3$.

Proof. Consider an edge $e = \{e_1, e_2\}$ of the MST. The *lune* is defined as the intersection of the two circles with centers at e_1 and e_2 and radius equal to $|e|$, see Figure A.1. There can be no other vertex of G inside the lune. For a contradiction assume that there is a

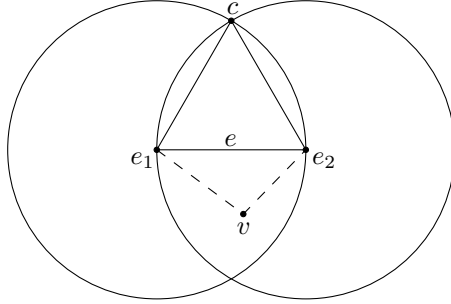


Figure A.1: There are no vertices inside the lune of an edge e of the MST

vertex v inside the lune. Then v is closer to both endpoints: $|ve_i| < |e|$ and so $|ve_i|^\alpha < |e|^\alpha$ for $i = 1, 2$ because $\alpha \geq 1$. In the cycle formed by the edges e , e_1v and ve_2 the edge e has the largest weight. By the cycle property e cannot be in the MST.

Since there can be no vertices inside the lune, an edge that makes the closest angle with e has an endpoint at one of the circle intersections. Let c be a point on one of these intersections. The triangle Δce_1e_2 is equilateral. All internal angles inside Δce_1e_2 are $\pi/3$. \square

A.2 Lower bound for the NISE-method

Theorem 5.6. For points in the plane there exists an instance such that

$$\text{weight}(T_{\text{NISE}})/\text{weight}(\text{MST}) = \Omega(8^\alpha).$$

Furthermore there exists an instance for $\alpha = 2$ such that $\text{weight}(T_{\text{NISE}})/\text{weight}(\text{MST}) = 5$.

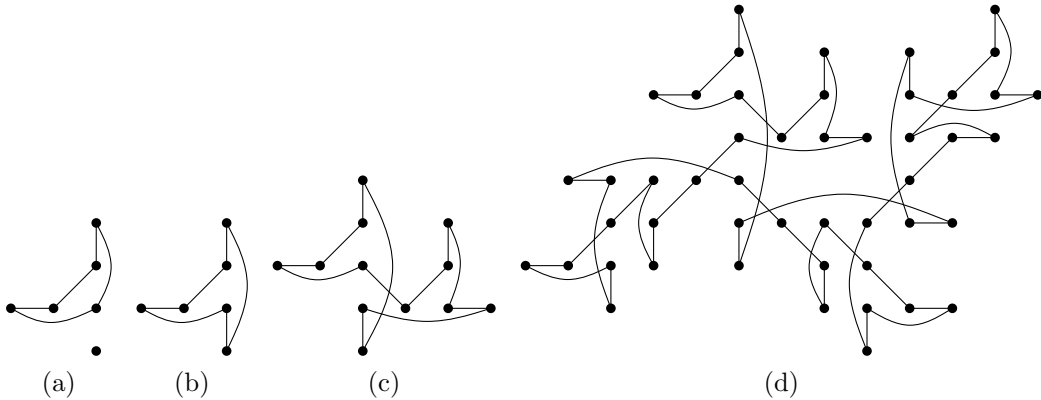


Figure A.2: Different steps of a construction that leads to a 6-shortcut.

Proof. The NISE-method can actually yield a bad result as the following example shows. The construction is depicted in Figure A.2. The points are placed on a regular grid with a minimum distance of 1 between two points. If the algorithm at some point has to make a choice between points that have the same distance to the tour we can assume for the purpose of the lower bound example that the algorithm makes a “wrong” choice. This does not mean that the algorithm can be improved in these tie-braker cases, because we can force the algorithm to make the wrong choices by changing some of the distances to $1 - \epsilon$ for a small number ϵ and get asymptotically the same results when ϵ tends to 0.

Figure A.2a shows the tour that can be the result of the NISE-algorithm for all but one of the points. When this last point is inserted in the tour the algorithm must take one of the 2-shortcuts out of the tour and insert a 3-shortcut. The NISE-method chooses to remove the shortest edge of the two, but since both have the same length we can assume that it removes the vertical-aligned 2-shortcut. The resulting tour is shown in Figure A.2b. The tour has one 3-shortcut where the two internal angles are π . We can repeat the whole construction so that we have two 3-shortcuts, next to each other in the tour. If we then insert a new point in the tour such that this point is closest to the point incident to the two 3-shortcuts the algorithm has to make a choice on which 3-shortcut is to be removed. The wrong choice leads to a 4-shortcut where the four vertices are on a straight line. See Figure A.2c. The construction can be repeated two more times which leads to a 6-shortcut where all six vertices are on straight line. See Figure A.2d. By then there is no more space in the plane to further repeat the construction.

We can make the construction somewhat worse by skewing parts of it such that all the 2-shortcuts which have an angle between the two edges of $\pi/2$ now have an angle of $2/3\pi$. This also leaves more room so that we can repeat the construction until we have an 8-shortcut. See Figure A.3. The minimum angle between edges is $\pi/3$ so that all the edges of the MST have unit length by Lemma A.1.

Let T_k be the tour after we’ve repeated the skewed construction k times and MST_k the corresponding MST. We can express the weight of the tour and the MST recursively. To create a k -shortcut we would first build up the construction for creating two $k-1$ -shortcuts. The two constructions are glued together with one 2-shortcut that has weight $\sqrt{3}^\alpha$, but this removes one edge of unit length from each construction. We then add one more point to it so that one of the $k-1$ -shortcuts is replaced by a k -shortcut and this adds another edge to the tour of unit length. For $\alpha = 2$ the weights of the tour and the MST can be expressed as follows:

$$\text{weight}(T_k) = \begin{cases} 6 & \text{if } k = 2, \\ 2 \cdot \text{weight}(T_{k-1}) + 2 + k^2 - (k-1)^2 & \text{if } k > 2. \end{cases}$$

$$\text{weight}(\text{MST}_k) = \begin{cases} 2 & \text{if } k = 2, \\ 2 \cdot \text{weight}(\text{MST}_{k-1}) + 1 & \text{if } k > 2. \end{cases}$$

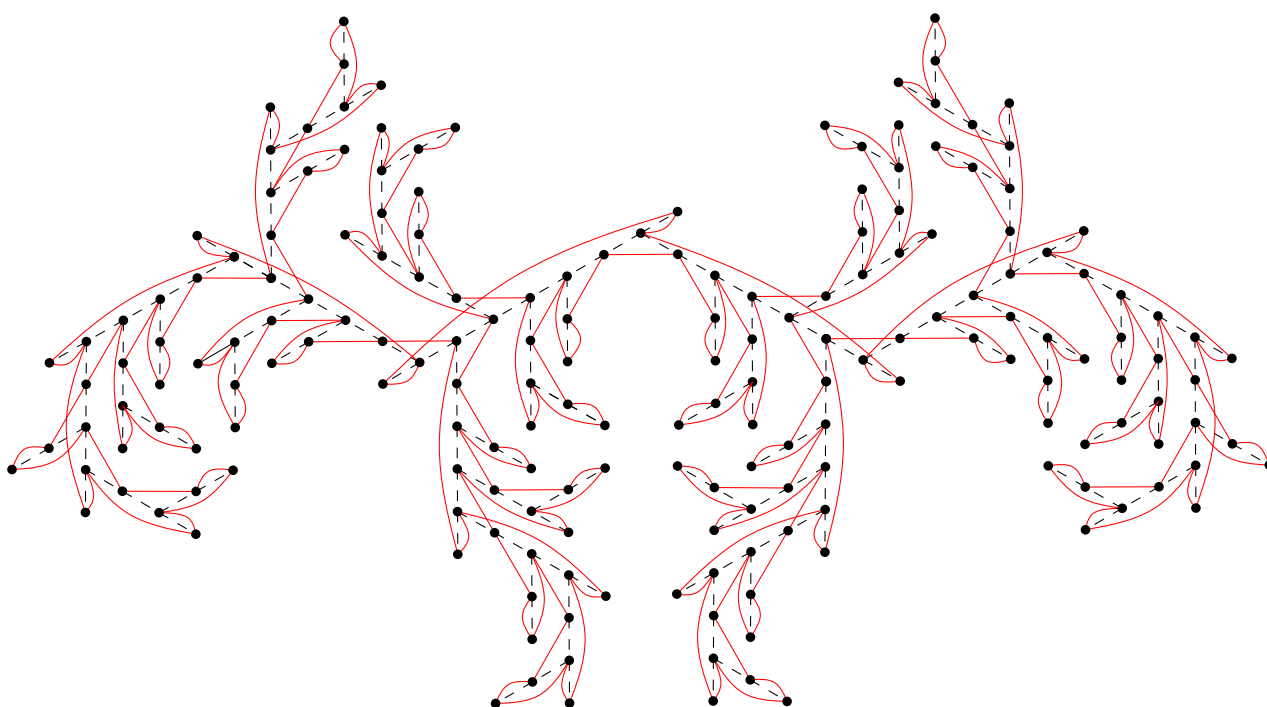


Figure A.3: Application of the NISE-algorithm to this point set can lead to an 8-shortcut in the final tour. The corresponding MST is drawn dashed.

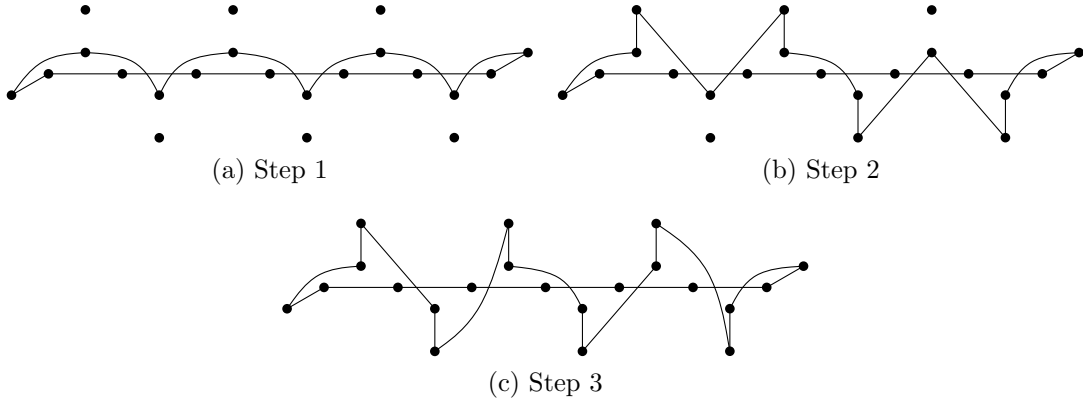


Figure A.4: If the NICE-algorithm processes the points in this order it could lead to long 4-shortcuts.

The two recurrences solve to:

$$\begin{aligned} \text{weight}(T_k) &= (15 \cdot 2^k - 8k - 20) / 4 \\ \text{weight}(\text{MST}_k) &= (3 \cdot 2^k - 1) / 4 \end{aligned}$$

For $\alpha = 2$ and $k = 8$ we get $\text{weight}(T_8)/\text{weight}(\text{MST}_8) = 939/191 \geq 4.91$. In Lemma 5.2 we did not make the assumption that the points are restricted to lie in the plane. This means that we can repeat the construction further if the points lie in a 3D-space and even further for higher dimensions. If we were given an infinite number of dimensions we could embed every step of our construction in a different dimension. If k approaches infinity then we have:

$$\lim_{k \rightarrow \infty} \frac{\text{weight}(T_k)}{\text{weight}(\text{MST}_k)} = \frac{1/4 (15 \cdot 2^k - 8k - 20)}{1/4 (3 \cdot 2^k - 1)} = 5.$$

Any MST of the points in the example of Figure A.3 use only edges of unit length. So the weight of the MST remains the same for all values of α . The tour includes one 8-shortcut. So we have that $\text{weight}(T_8)/\text{weight}(\text{MST}_8) = \Omega(8^\alpha)$. \square

A.3 Lower bound for the NICE-method

For the NICE-method we can also give examples on which the method performs bad, but not as bad as with the NISE-method.

Lemma A.2. *For arbitrary large n_0 , there exists a set of $n \geq n_0$ points in the plane such that $\text{weight}(T_{\text{NICE}})/\text{weight}(\text{MST}) = \Omega(\sqrt{8 + 3\sqrt{2}}^\alpha) = \Omega(3.49^\alpha)$.*

Proof. The construction is depicted in Figure A.4. All edges of the MST have unit length and all angles between two edges of the MST that are incident to each other and share an endpoint make an angle of either $2/3\pi$ or π . If at some point two or more points have the minimum distance to the tour we may choose how the algorithm proceeds in the worst case. We've defined $\text{cost}(j, p)$ as the increase in the length of the tour if we insert point p between v_j and v_{j+1} . That means we have:

$$\text{cost}(j-1, p) = |v_j p|^\alpha + |v_{j-1} p|^\alpha - |v_{j-1} v_j|^\alpha.$$

When $|v_{j-1} p|^\alpha - |v_{j-1} v_j|^\alpha = |v_j p|^\alpha - |v_j v_{j+1}|^\alpha$ this implies $\text{cost}(j-1, p) = \text{cost}(j, p)$ and we can assume that the algorithm makes the worst possible choice. Our construction consist of three steps.

Step 1: The algorithm starts with the leftmost point and inserts all the points on the zig-zag line of the MST in order from left to right. This creates a series of 2-shortcuts as in Figure A.4a.

Step 2: First consider what happens when we insert one of the points, say p , that has not been inserted yet in the tour. The closest distance from p to the tour is to a vertex v_j where both edges $\{v_j, v_{j-1}\}$ and $\{v_j, v_{j+1}\}$ are 2-shortcuts with the same weight. Hence $\text{cost}(j-1, p) = \text{cost}(j, p)$ and we can choose how the algorithm proceeds. In both cases a 3-shortcut is generated.

We group the points that have not been inserted in the tour in groups of three points. Of every group we insert the first and the third point in the tour such that the resulting 3-shortcuts are incident to the same point. See Figure A.4b.

Step 3: Now we insert the second point in every group of three. This creates 4-shortcuts. See Figure A.4c.

Finally we can replace every edge that is still in the tour with a series of points whose distances follow a geometric series with common ratio equal to $1/2$ according to Lemma 4.18. We repeat the pattern n times. The pattern includes:

- Three series of 2-shortcuts of weight $\frac{(3/2)^\alpha}{1-2^{-\alpha}} = 3^\alpha / (2^\alpha - 1)$.
- Three 2-shortcuts of weight $\sqrt{3}^\alpha$.
- One 2-shortcut of weight 2^α .
- One 3-shortcut of weight $\sqrt{7}^\alpha$.
- One 4-shortcut of weight $\sqrt{12}^\alpha$.

At one end we have a 2-shortcut of weight 2^α and a 1-shortcut of weight 1. At the other hand we have a 1-shortcut of weight 1. The weight of the MST part of one serie of points whose distance follow a geometric series is $1/(1 - (1/2)^\alpha) = 1 + 1/(2^\alpha - 1)$.

If n approaches infinity we have:

$$\begin{aligned} \frac{\text{weight}(T_{\text{NICE}})}{\text{weight}(\text{MST})} &= \lim_{n \rightarrow \infty} \frac{2^\alpha + 2 + n \left(3^{\alpha+1}/(2^\alpha - 1) + 3\sqrt{3}^\alpha + 2^\alpha + \sqrt{7}^\alpha + \sqrt{12}^\alpha \right)}{3 + n(3/(2^\alpha - 1) + 9)} \\ &= \frac{3^{\alpha+1}/(2^\alpha - 1) + 3\sqrt{3}^\alpha + 2^\alpha + \sqrt{7}^\alpha + \sqrt{12}^\alpha}{3/(2^\alpha - 1) + 9} \end{aligned} \quad (\text{A.4})$$

The expression in (A.4) is more than 2^α for $\alpha \geq 1.84$.

For larger values of α it is better to use smaller angles than $2/3\pi$ in the zig-zag pattern. For our construction to work these angles must be at least π . If the angle is π then the length of the three 2-shortcuts becomes $\sqrt{2}$, the length of the 3-shortcut becomes $\sqrt{5 + 2\sqrt{2}}$ and the length of the 4-shortcut becomes $\sqrt{8 + 3\sqrt{2}}$. We then have:

$$\frac{\text{weight}(T_{\text{NICE}})}{\text{weight}(\text{MST})} = \frac{3^{\alpha+1}/(2^\alpha - 1) + 3\sqrt{2}^\alpha + 2^\alpha + \sqrt{5 + 2\sqrt{2}}^\alpha + \sqrt{8 + 3\sqrt{2}}^\alpha}{3/(2^\alpha - 1) + 9} \quad (\text{A.5})$$

The expression in (A.5) is more than the one in (A.4) for $\alpha \geq 3.7$. □

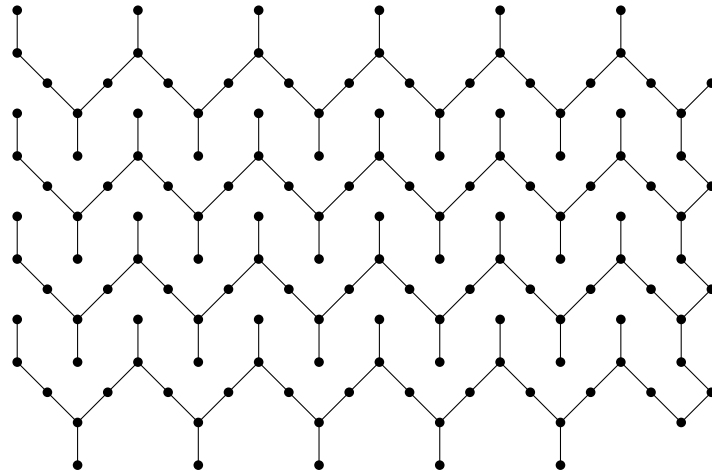
We can extend the construction of Lemma A.2 to give a lower bound example compared to the optimum tour for TSP^α .

Theorem 5.7. The approximation factor of the NICE-algorithm is

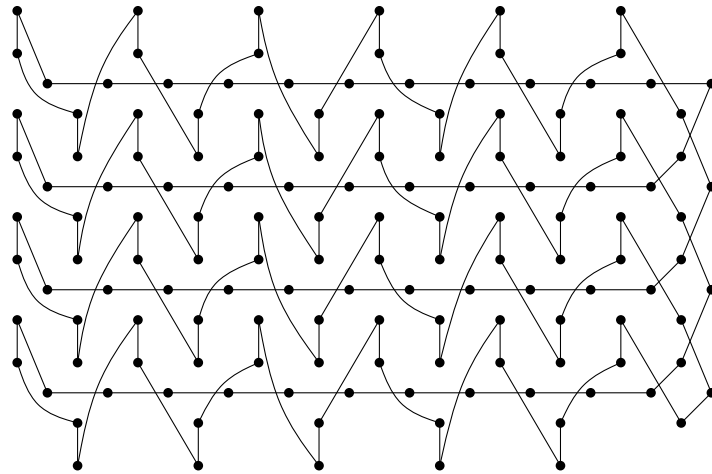
$$\sqrt{8 + 3\sqrt{2}}^\alpha = \Omega(3.49^\alpha).$$

Proof. The construction is depicted in Figure A.5. We repeat the pattern of Lemma A.2 n times per row and repeat this for m rows in total. We've drawn the zig-zag patterns with an angle of π , but we could also have used angles of $2/3\pi$ as in Lemma A.2. The closest distances in a row are still 1. The distance from each row to the one above or below is also 1. The NICE-algorithm chooses to insert at each iteration the point that is closest to one of the points already in the tour. In our construction there are often multiple points that are closest. In this case we can assume that the algorithm makes a certain choice. We assume that the algorithm processes the points row for row in the same order as in Lemma A.4 so that we get the same tour for each row. When the algorithm has inserted all points on one row it starts with the rightmost point from the next row. The resulting tour is shown in Figure A.5b. The corresponding MST is shown in Figure A.5a. All edges of the MST have unit length and thus also unit weight.

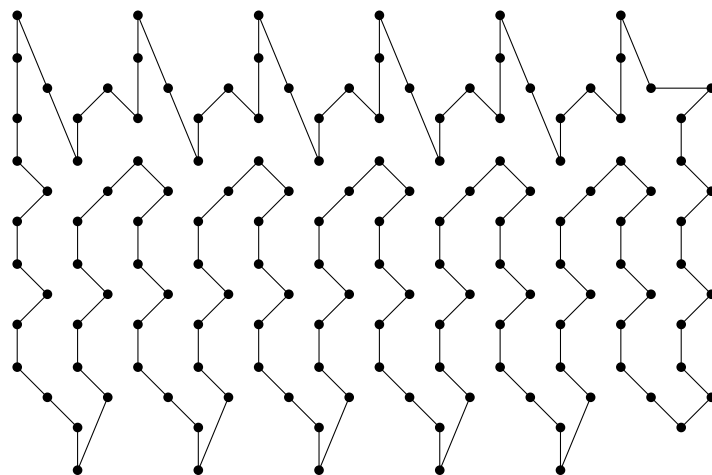
We assume that both n and m approach infinity. An optimal tour can pass through all the points in a vertical zig-zag pattern as shown in Figure A.5c. All edges in the middle are of unit length. So the optimal tour uses $9n(m - 2)$ edges of unit length plus some longer edges near the top and bottom boundaries. We denote the upper boundary of the



(a) MST



(b) Tour produced by the NICE-algorithm.



(c) Optimal TSP^α tour.

Figure A.5: Lower bound instance for the NICE-algorithm.

optimal tour with $\text{OPT}_U(n)$ and the lower boundary with $\text{OPT}_D(n)$. Then we have that: $\text{weight}(T_{\text{OPT}}) = 9n(m-2) + \text{weight}(\text{OPT}_U(n)) + \text{weight}(\text{OPT}_D(n))$.

The length of the NICE-tour consist of $(n-1)m$ times the pattern of Lemma A.2 plus some extras on the left and right boundaries. Denote the shortcuts on the left boundary with $T_L(m)$ and the shortcuts on the right boundary with $T_R(m)$. By Lemma A.2 we have that:

$$\begin{aligned} \text{weight}(T_{\text{NICE}}) &= (n-1)m \left(3 + 3\sqrt{2}^\alpha + 2^\alpha + \sqrt{5 + 2\sqrt{2}}^\alpha + \sqrt{8 + 3\sqrt{2}}^\alpha \right) \\ &\quad + \text{weight}(T_L(m)) + \text{weight}(T_R(m)). \end{aligned}$$

Taking it together we get:

$$\begin{aligned} &\frac{\text{weight}(T_{\text{NICE}})}{\text{weight}(T_{\text{OPT}})} \\ &= \lim_{n,m \rightarrow \infty} \frac{(n-1)m \left(3 + 3\sqrt{2}^\alpha + 2^\alpha + \sqrt{5 + 2\sqrt{2}}^\alpha + \sqrt{8 + 3\sqrt{2}}^\alpha \right) + \text{weight}(T_L(m)) + \text{weight}(T_R(m))}{9n(m-2) + \text{weight}(\text{OPT}_U(n)) + \text{weight}(\text{OPT}_D(n))} \\ &= \frac{3 + 3\sqrt{2}^\alpha + 2^\alpha + \sqrt{5 + 2\sqrt{2}}^\alpha + \sqrt{8 + 3\sqrt{2}}^\alpha}{9} \end{aligned} \tag{A.6}$$

The highest order term in (A.6) is $\sqrt{8 + 3\sqrt{2}}^\alpha$ which is more than 3.49^α . \square

A.4 Lower bound for the NI-algorithm

Theorem 5.8. There exists a set of points such that

$$\text{weight}(T_{\text{NI}})/\text{weight}(\text{MST}) = \Omega \left(\sqrt{5 + 2\sqrt{2}}^\alpha \right) = \Omega(2.79^\alpha).$$

Proof. The construction is shown in Figure A.6 and is similar to the one of Theorem 5.7. The pattern in Figure A.6 can be repeated n times. All edges of the MST have unit length. The construction consist of three steps.

- Step 1: The algorithm inserts all points on the zig-zag line from left to right. This yields a tour that consists of only 2-shortcuts and two 1-shortcuts at the ends. See Figure A.6a.
- Step 2: The algorithm inserts all points that have distance one to the zig-zag line. The NI-algorithm inserts these points in the cheapest way. We consider the insertion of one of these points, namely p . Denote the closest points on the zig-zag

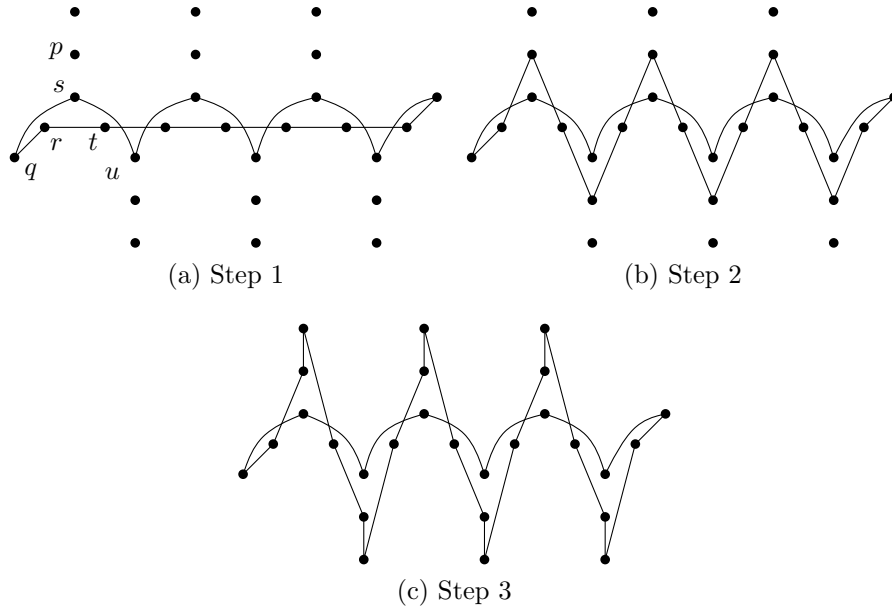


Figure A.6: If the NI-algorithm processes the points in this order it could lead to long 3-shortcuts.

line with q, r, s, t and u as in Figure A.6a. By the topology of the construction there are three options to consider. The algorithm could insert p between q and s , between s and u or between r and t . Define $\text{cost}(x, y, z)$ to be the increase in the weight of the tour if point x is inserted between y and z . Thus we know that $\text{cost}(x, y, z) = |xy|^\alpha + |xz|^\alpha - |yz|^\alpha$. By symmetry it follows that $\text{cost}(p, q, s) = \text{cost}(p, s, u)$. We consider only the first two options.

$$\begin{aligned}
\text{cost}(p, q, s) &= |pq|^\alpha + |ps|^\alpha - |qs|^\alpha \\
&= \sqrt{2^2 + 1^2 - 4 \cos \frac{3\pi}{4}} + 1 - 2^\alpha \\
&= \sqrt{5 + 2\sqrt{2}}^\alpha + 1 - 2^\alpha \\
\text{cost}(p, r, t) &= |pr|^\alpha + |pt|^\alpha - |rt|^\alpha \\
&= 2|pr|^\alpha - |rt|^\alpha \\
&= 2\sqrt{2 - 2 \cos \frac{3\pi}{4}} - \sqrt{2}^\alpha \\
&= 2\sqrt{2 + \sqrt{2}}^\alpha - \sqrt{2}^\alpha
\end{aligned}$$

For $\alpha \geq 2$ it follows that $\text{cost}(p, r, t) \leq \text{cost}(p, q, s)$. So the algorithm inserts p in the worst case between r and t . The other points at distance 1 from the zig-zag line are inserted in the same way. This leads to the tour in Figure A.6b.

- Step 3: The algorithm inserts all points at distance 2 from the zig-zag line. This introduces 3-shortcuts as shown in Figure A.6c.

Finally we can replace every edge that is still in the tour with a series of points whose distances follow a geometric series with common ratio equal to $1/2$ according to Lemma 4.18. The pattern is repeated n times. The pattern includes:

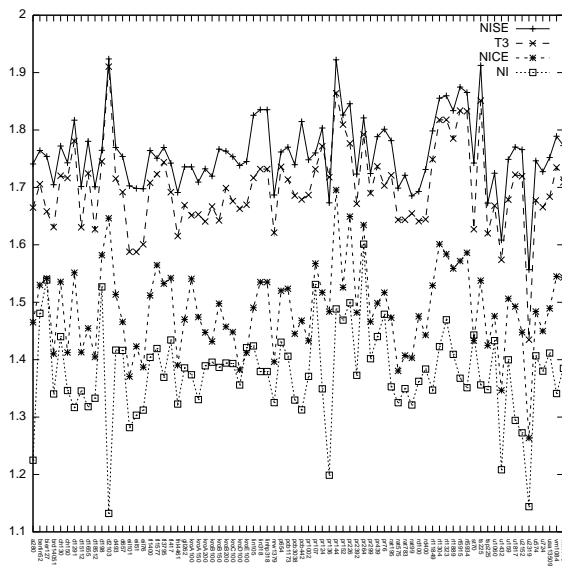
- A 2-shortcut of weight $\sqrt{2 + \sqrt{2}^\alpha}$.
- A serie of 2-shortcuts of total weight $3^\alpha / (2^\alpha - 1)$.
- A 3-shortcut of weight $\sqrt{5 + 2\sqrt{2}^\alpha}$.
- A 2-shortcut of weight 2^α .

At one end of the zig-zag line we have a 1-shortcut of weight 1 and at the other hand a 2-shortcut of weight 2^α and a one shortcut of weight 1. The weight of the MST part of one serie of points whose distance follow a geometric serie is $1/(2^\alpha - 1)$. If n approaches infinity we have:

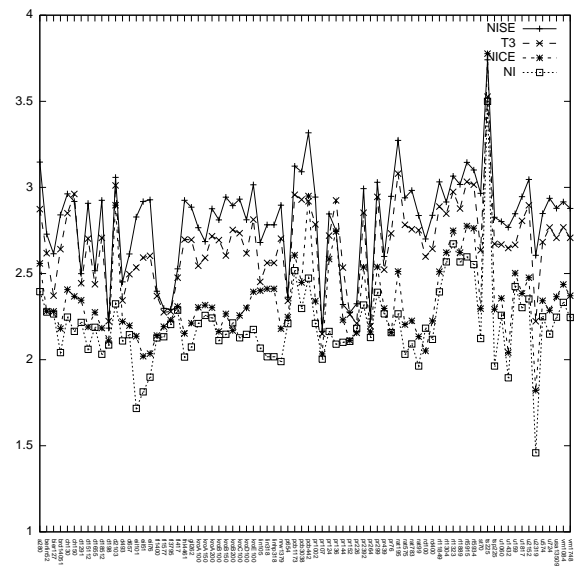
$$\begin{aligned} \frac{\text{weight}(T_{\text{NI}})}{\text{weight}(\text{MST})} &= \lim_{n \rightarrow \infty} \frac{2 + 2^\alpha + n(\sqrt{2 + \sqrt{2}^\alpha} + 3^\alpha / (2^\alpha - 1) + \sqrt{5 + 2\sqrt{2}^\alpha} + 2^\alpha)}{2 + n(3 + 1/(2^\alpha - 1))} \\ &= \frac{\sqrt{2 + \sqrt{2}^\alpha} + 3^\alpha / (2^\alpha - 1) + \sqrt{5 + 2\sqrt{2}^\alpha} + 2^\alpha}{3 + 1/(2^\alpha - 1)} \end{aligned} \quad (\text{A.7})$$

The expression in (A.7) is more than 2^α for $\alpha \geq 2$. □

A.5 Results of the experiments on TSPLIB instances

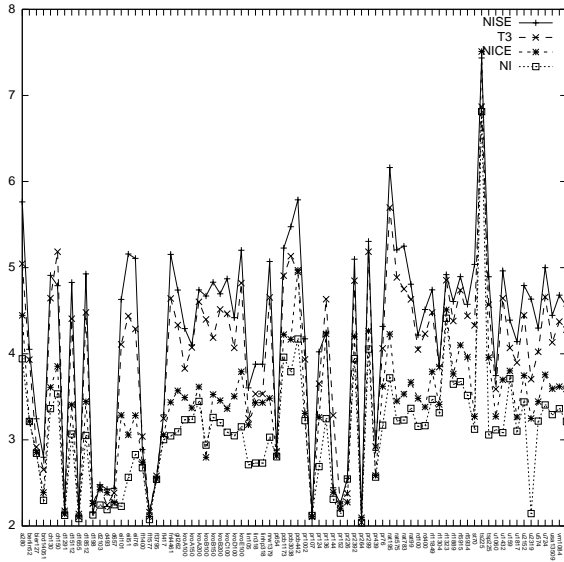


(a) $\alpha = 1$

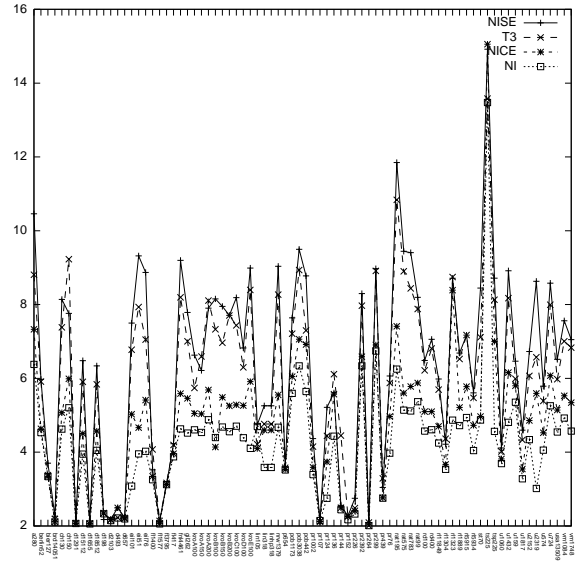


(b) $\alpha = 2$

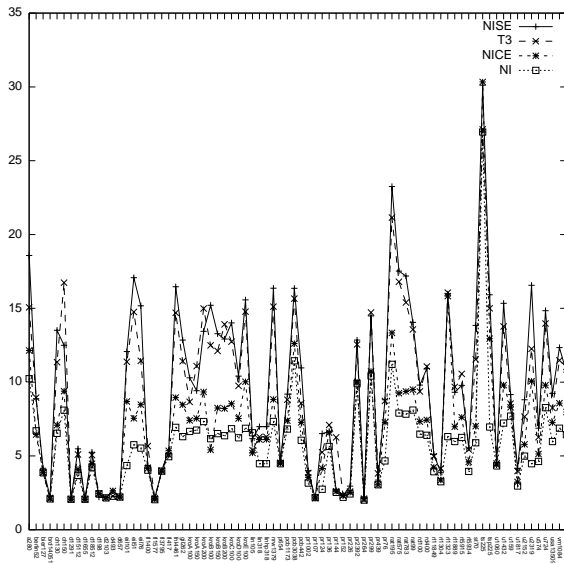
Figure A.7: The ratio $\text{weight}(T)/\text{weight}(\text{MST})$ for the different algorithms applied to instances from TSPLIB.



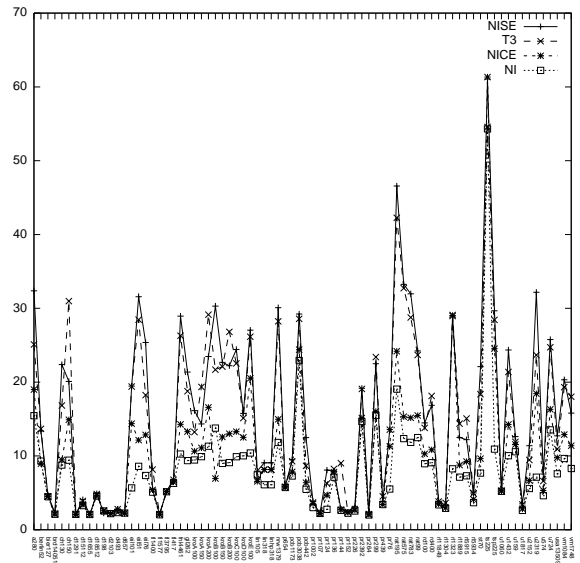
(c) $\alpha = 3$



(d) $\alpha = 4$

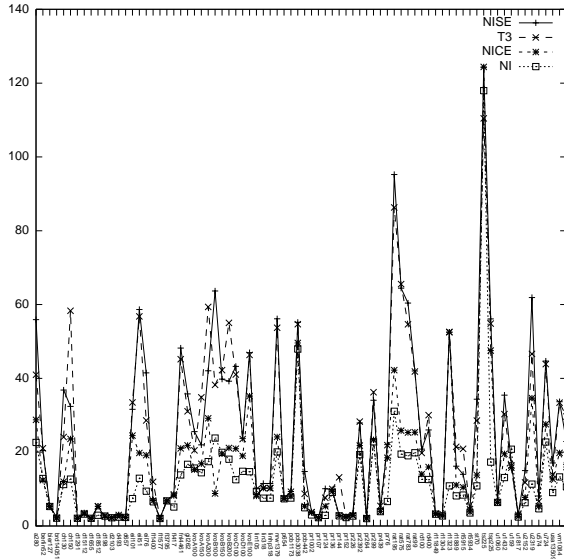


(e) $\alpha = 5$

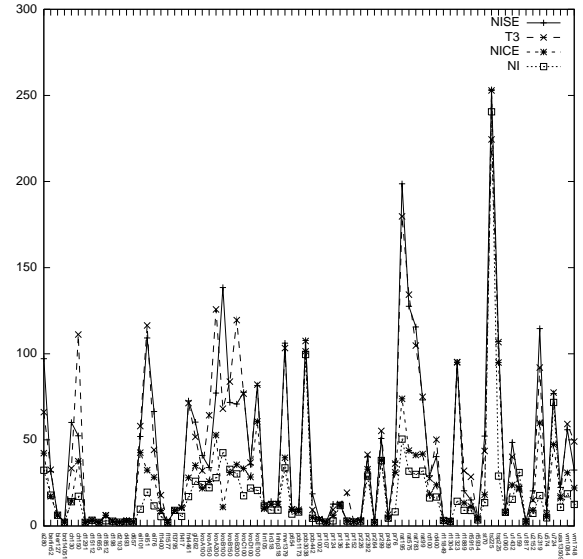


(f) $\alpha = 6$

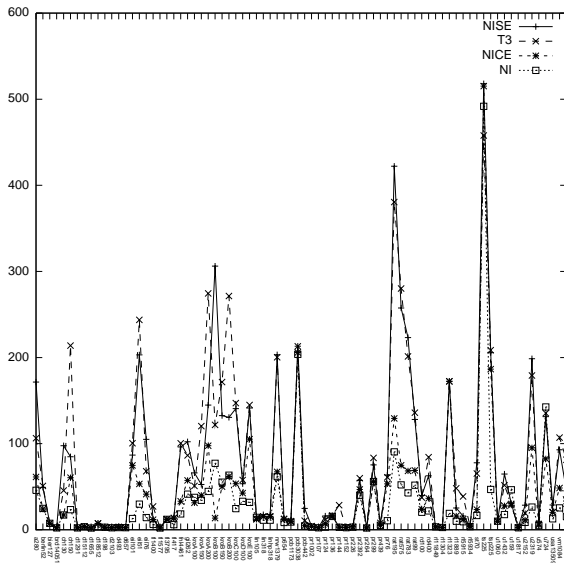
Figure A.7: The ratio $\text{weight}(T)/\text{weight}(\text{MST})$ for the different algorithms applied to instances from TSPLIB.



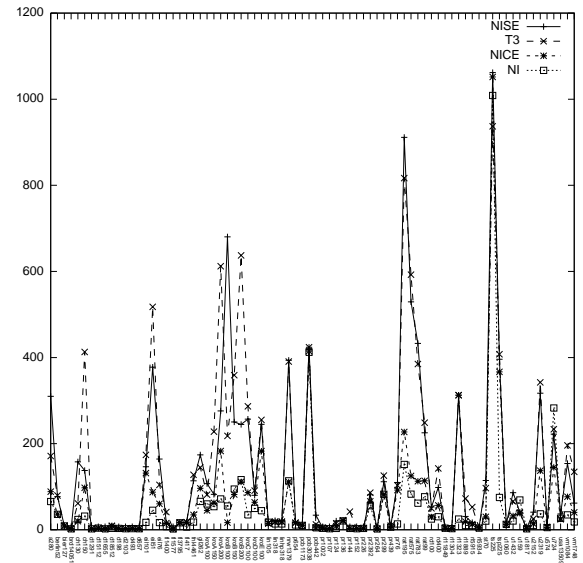
(g) $\alpha = 7$



(h) $\alpha = 8$



(i) $\alpha = 9$



(j) $\alpha = 10$

Figure A.7: The ratio $\text{weight}(T)/\text{weight}(\text{MST})$ for the different algorithms applied to instances from TSPLIB.