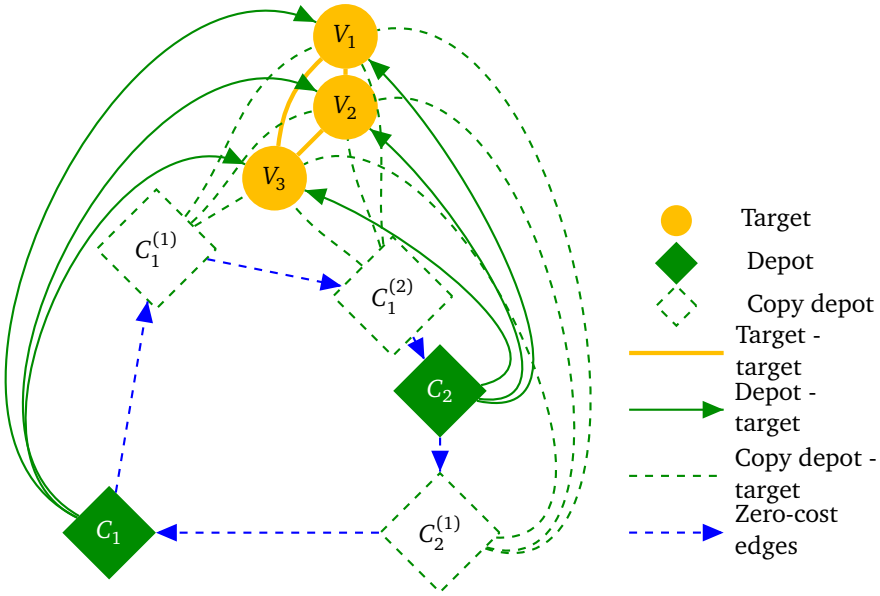
OPTIMAL ROUTING OF CABLES IN ORDER TO UPGRADE THE COPPER NETWORK

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Mathematical Sciences at Utrecht University



Annelies Jonker, student number: 3901211
January, 2019

Universiteit Utrecht

kpn

supervision:
prof. dr. R.H. Bisseling (UU)
dr. R. Hillen (KPN)

Abstract

In the past two decades, the global internet traffic volume has grown exponentially and it is expected to grow in the future. The copper cable network of KPN will not be able to handle the expected demand and the optic fibre network will not be extensive enough in time. Therefore, the current copper cable network needs an upgrade. This is done by installing optic fibre cables from street cabinets to the network. Then, customers experience a higher data transmission speed as the length of the copper cable is reduced.

Digging trenches to install these cables is a very costly process. To minimise these costs, and to minimise the risk of failure of the network, a heuristic method based on the Lin-Kernighan algorithm is developed that solves an asymmetric, precedence-constrained Travelling Salesman Problem. However, if an increased risk of failure is allowed, a further cost reduction can be made. A branch-and-bound method has been used to investigate this reduction. An educated choice can then be made, whether the cost decrease is in balance with the increased failure risk.

Acknowledgements

This research is executed at the Data and Analytics department at KPN, Amersfoort. I want to thank KPN for giving me the oppurtunity to write my master's thesis with them, and Robert Hillen in particular for his insights. I thank my supervisor Rob Bisseling from the Mathematics department at Utrecht University for his valuable comments. Moreover, I want to thank Michiel Baatsen for his input and his moral support. Last but not least, I want to thank my parents for their continuous support.

Contents

Chapter 1

Introduction

The demand for internet bandwidth has increased enormously in the past years. Figure 1.1 shows the growth of the global internet traffic volume in PetaBytes per month on a linear scale (left) and on a logarithmic scale (right), which has been exponential between the years 2000 and 2006. Even though a slight decay is shown in the growth of traffic volume after 2006, the traffic volume is predicted to grow for two main reasons: the internet is becoming a popular way to consume (High Definition) video, and optical fibre and other high bandwidth connections are more and more accessible to customers [13]. Telecom companies need to invest massively to keep up with the demand for bandwidth in the future. Therefore, in order to remain a competitive partner in the telecom industry, KPN needs to invest in her network.

Most of the current internet network consists of copper cables, over which a maximum speed of data transmission of 400 Mbit/s is possible. Several techniques exist to increase the speed of data transmission over these cables, however, the available speed at home is mostly limited by the length of the cable.

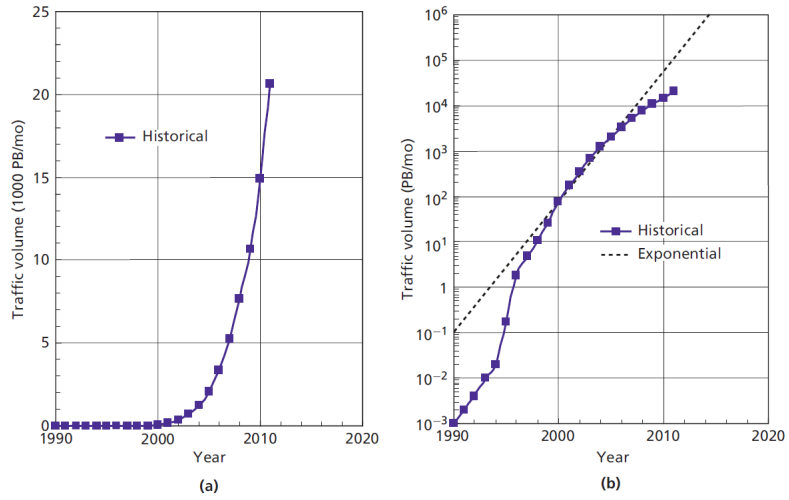


Figure 1.1 – Explosive growth rate of the traffic volume over the internet, expressed in PetaBytes per month. The left figure shows the growth on a linear scale, the right figure on a logarithmic scale. Figure from [10].

Beside the copper network, KPN has an optical fiber network which currently services approximately 25% of all households in the Netherlands. Such fiber connections are

referred to as fiber to the home (FttH). These cables currently allow a typical up- and download speed of 500 Mbit/s; however in the future they will allow a speed of 100 Gbit/s. Therefore, KPN will invest in expanding the current optical fibre network. However, the expansion of the optical fiber network to produce more FttH connections is a very slow and costly process. Hence the need to expand and upgrade the current copper cable network.

The background of KPN is described in Section 1.1. The current network of KPN is discussed in Section 1.2 and Section 1.3 describes the problem that this thesis focuses on. Finally, Section 1.4 gives the structure of the thesis.

1.1 Company background

KPN is the owner of the Dutch fixed-line telephone network and has around 13 thousand employees. The company offers fixed-line and mobile telephony, internet and TV services to all customers and provides IT-solutions to corporate customers.

KPN started as a public company in 1915, which was called P&T (Staatsbedrijf der Posterijen en Telegrafie) and changed its name to PTT (Staatsbedrijf der Posterijen, Telegrafie en Telefonie) in 1928. This company hosted telecommunication services through telephones and telegraphs, and hosted the postal service. The Dutch government decided to privatise PTT in 1989, and KPN (Koninklijke PTT Nederland N.V.) was founded. The most important divisions of KPN were PTT Telecom and PTT Post. In 1998, these divisions split ways and PTT Telecom continues under the name KPN.

As from 2011, KPN needs to invest in its current network to keep up with the growing amount of data traffic. It does so by expanding the optic fibre network and upgrading its copper network. However, this is a costly (and slow) process that will take many years. One of the goals of KPN is to become the best service provider in The Netherlands.

1.2 Understanding the network

This section gives an overview of the hierarchical structure of the current network, from all homes that use the internet to the international network.

At the lowest level of the network, households are connected to the internet through street cabinets. Depending on the type of cable, households are either connected to a KVD[1] via copper cables or to a POP (Point of Presence) cabinet via optical fibre cables.

These street cabinets can be connected to the network on two levels. See Figure 1.2 for a visualisation. Multiple street cabinets, both KVD and POP locations, are connected to a MetroAcces (MA) location by a PAN[2] cable. In turn, a SAN[3] cable connects multiple street cabinets to a PAN cable. PAN and SAN cables are connected in a ring

[1] for *kabelverdeelkast* in Dutch
[2] for *primaire deel van het aansluitnet* in Dutch
[3] for *secundaire deel van het aansluitnet* in Dutch

structure, also referred to as a redundant connection, to minimise the risk of failure of part of the network: if a cable between two street cabinets is broken, all cabinets will still be working properly. An MA location is the endpoint of the local networks and transports the data to a higher level in the network.

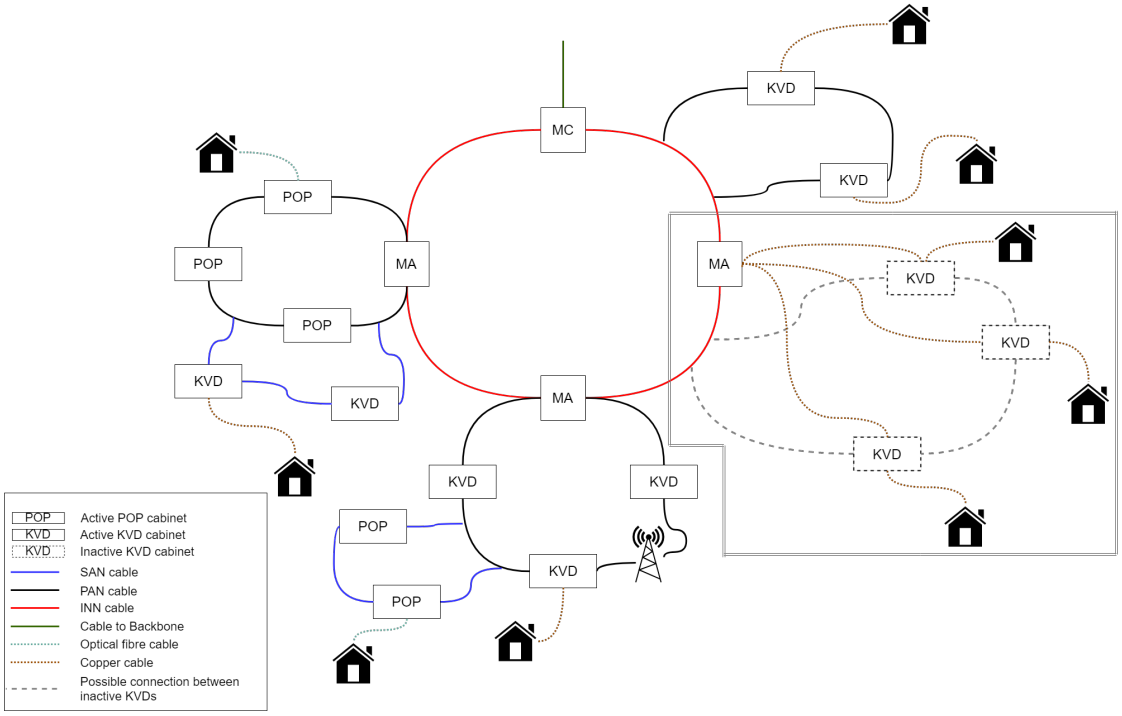


Figure 1.2 – The local networks that communicate with an MA location, which in turn communicate with an MC location.

Several MA locations are connected to one of the MetroCore (MC) locations in the Netherlands through an Internet Node Net (INN) cable. To reduce the risk of network failure, MA locations are connected in a ring structure. An MC location severely compresses the data and transports it to a higher level in the network.

Multiple MC locations are connected to one of the two independent backbone (BB) networks. The independence ensures that broken links in one of the networks will not cause a national network failure. Finally, the BB is connected to 'the net', the international network. An overview of the various levels of the KPN network is given in Figure 1.3.

A distinction is made between active and inactive KVDs. An active KVD connects the copper cable that runs from the household to the KVD to the optical fibre cable (PAN, SAN or INN) that runs from the MA location to the KVD. An active KVD contains multiple DSLAMs, which are machines that facilitate the connection of the copper to the optical fibre cable. An inactive KVD does not contain DSLAMs. The inactive KVD is connected to the household and to the MA location by a copper cable, as shown in Figure 1.2.

Finally, an MA-area is the area that contains the local network connected to one MA location. An MC-area contains the local network and all MA-locations that are connected to one MC location.

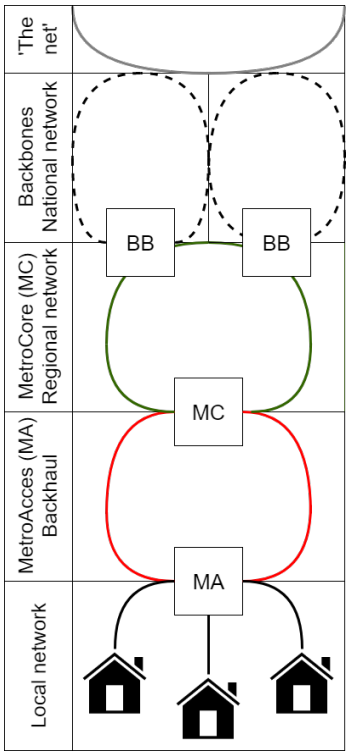


Figure 1.3 – Overview of the different levels of the KPN network.

1.3 Problem statement

As mentioned before, the disadvantage of using copper cables is that the speed for data transmission is limited by the length of the cable. The maximum acceptable length is approximately 100 metres: the signal will be too distorted and too weak when the data has to travel a longer distance. Optical fibre cables don't have that disadvantage. However, the telephone network consists of copper cables and KPN uses those cables for internet, therefore almost all households have a copper cable connection but not necessarily an optical fibre cable connection.

Speed of data transmission for the households that are connected to an inactive KVD can be improved by activating it. This is done by installing an optical fibre cable from a PAN, SAN or INN cable to the KVD. The main limitation of the speed of data transmission for households connected to this KVD is now only the length of the copper cable from the KVD to the household. Also, a DSLAM must be installed to activate the KVD. As the problem only concerns KVDs and not POP locations, the term *street cabinet* is used for KVDs in the remainder of this thesis.

KPN intends to upgrade its current copper network by activating 2200 KVDs in the year 2019. By far, the most expensive part of activating a street cabinet is the cost of digging trenches for the cables, so it is important to choose an optimal route to minimise the distance that needs to be dug. Another objective is to reduce the amount of work for the engineers: they plan routes by hand now. Moreover, at this time there is no accurate approximation procedure for estimating the total cost of digging trenches for 2200 cabinets since the distance that needs to be dug is unknown.

To reduce the risk of network failure, the cables connecting the currently inactive cabinets must be connected in a ring structure. However, when less than 2000 customers are subject to a KVD, a cable to and from the KVD in the same trench is allowed. Such a trench is called a SPoF (Single Point of Failure) and such a connection is called non-redundant or single-sided.

The trenches need to be dug along existing paths, roads and railroad tracks. The inactive street cabinets can be connected to INN, PAN and SAN cables (see Figure 1.2). However, the newly installed cable that connects one or more cabinets should end at the same type of cable in the existing network as it starts from. Moreover, MC areas are independent, which means that a cable cannot connect one end of the new cable in one area and connect the other end in a different MC area. This yields approximately 150 independent problems.

However, these problems contain many street cabinets, so smaller problems are investigated first. Therefore, the problems that are considered in this thesis are on a lower level in the network. The problems concern MA areas. Even though it is allowed to connect street cabinets by a cable that runs through multiple MA areas, this is ignored to advance an upscaling to an MC area.

Finally, KPN does not allow connections from a KVD to an INN cable in urban areas. The definition of 'urban' is taken from Statistics Netherlands (CBS) and is defined as follows [3]. Let x be the number of addresses per km^2, calculated on a square of 500×500 m^2. Then x is a measure for how urban an area is. Connections to INN cables are allowed for $x \leq 1500$.

1.4 Structure of the thesis

Chapter 3 gives a mathematical formulation of the problem described above. Then, in Chapter 4 a heuristic model is described for obtaining tours that connect a set of street cabinets to cables. An exact method to solve the same problem is given in Chapter 5, which is used to examine the quality of the heuristic model in Chapter 7. Connections where multiple cables are allowed in the same trench are handled in Chapter 6. Results and conclusions of this research, and also directions for further research, are given in Chapter 7 and Chapter 8.

Chapter 2

Literature review

The problem described in Section 1.3 is closely related to the Travelling Salesman Problem (TSP), which is as follows: find a route of minimal distance where a salesman travels through n targets, through each exactly once, and returns to its initial city (depot). Another formulation is: find a minimal distance Hamiltonian circuit through a given vertex set, i.e. a circuit of minimum distance where each vertex is passed through exactly once.

The TSP is an instance of the Generalised Multiple Depot, Multiple Travelling Salesmen Problem (GMTSP), where several salesmen are located at multiple distinct depots. The objective is to minimise the total distance travelled, where all targets are visited exactly once. Not all salesmen must be deployed. The Multiple Depot, Multiple Travelling Salesman Problem (MDMTSP) is an instance of the GMTSP, where all salesmen are necessarily deployed.

The problem described in the problem statement is an instance of the GMTSP. The TSP is known to be an NP-hard problem, hence the GMTSP is NP-hard as well. Therefore, heuristic algorithms are used to find acceptable, possibly non-optimal, solutions. Malik et al. [14] propose an algorithm for a GMTSP where each target is visited by at least one salesman and where costs satisfy the triangle inequality. They provide an α-approximation algorithm, which means the algorithm runs in polynomial time and the solution cost is at most α times the cost of the optimal solution, for $\alpha = 2$. According to the author, α-approximation algorithms for $\alpha < 2$ are not known for the GMTSP.

However, some work has been done on transforming the MDMTSP to a standard TSP. After the transformation, one of the many algorithms available for the TSP can be used to find a Hamiltonian circuit in the graph. GuoXing [7] proposes a transformation for the MDMTSP, where salesmen do not have to return to their original depot and where each salesman visits a target exactly once, to the standard TSP. Oberlin et al. [16] consider a MDMTSP, where each target is visited at least by one salesman and where each salesman returns to its initial depot, and show how to transform it to an asymmetric TSP. Some effective exact and heuristic algorithms for the asymetric TSP are discussed below.

2.1 Exact algorithms for the TSP

Finding the optimal route by checking all possible paths for a problem of n targets requires an exhaustive search of $(n-1)!$ possibilities: any target can be chosen as starting point. Then there are $n-1$ possibilities for the second target, $n-2$ possiblities for the third target, etcetera. A direct search would take a time proportional to the number of possible paths which requires unacceptable computation time, even for small n. Therefore, several exact algorithms have been designed. They can be divided into three types: Dynamic Programming, branch-and-bound algorithms and linear programming (LP). Each of them is discussed below.

The idea of Dynamic Programming is that, after a number of targets have been visited, the optimal tour through all n targets contains the optimal path through the remaining subset. However, the drawback of this method is that it is restricted to small problems due to the required computation time, being proportional to $n^2 2^n$ [1].

Branch-and-bound algorithms partition the set of all feasible solutions into smaller subsets and systematically evaluate them until the optimal solution is found. These algorithms can be used on LP problems, but are not restricted to them. Relaxation of one of the constraints of an LP problem gives lower bounds on the cost of the optimal solution and are a guide for the next branching step. The drawback of this method used to be the weak lower bound. Held and Karp [8] greatly improved this method: they obtained a much stronger lower bound (Held-Karp bound) by adding subtour constraints to the LP, meaning no isolated tours through a subset of n targets are allowed. Held and Karp found the lower bound by constructing a 1-tree; i.e. find the cheapest two edges connecting target v (which they call target 1) and add this cost to the cost of the minimum spanning tree through the remaining targets. This method converges to the best possible lower bound. The time it takes for the method to converge is unsatisfactory, however a good bound can be obtained by an acceptable number of runs [1].

2.2 Heuristic algorithms for the TSP

Even though several exact algorithms exist for the TSP, they are usually time-consuming. Therefore, heuristic algorithms that have led to great results in the past are also considered.

The famous Lin-Kernighan (LK) algorithm, first introduced in [12] and named after its authors, has been a milestone in research on the TSP. The algorithm is a tour improvement method, which requires a feasible initial tour and then attempts to improve it by using heuristics. Improvements are conducted by k-opt moves. A k-opt move removes k edges and replaces them by another set of k edges that results in a feasible tour of lower cost. In the past, optimal solutions have been obtained by the LK algorithm for small problems, up to 100 cities [1].

Since then, the algorithm has been improved and modified multiple times. Martin et al. [15] introduced the Chained Lin-Kernighan algorithm, which has produced solutions within 1% of optimal tours and optimal tours for problems of large scale. How-

ever, for modest-size instances (up to 1000 cities), the algorithm fails to produce optimal tours [1].

Furthermore, Helsgaun [9] has played a big role in improving the Lin-Kernighan algorithm by a set of modifications which he introduced in his LKH method. He tested his algorithm on problems that were in the TSPLIB library: a database providing researchers with various TSP [17]. He reports optimal solutions being found for all TSPLIB instances that were present in the year 2000, where the largest instance contained 13,509 cities.

Chapter 3

Mathematical formulation of the problem

The problem stated in Section 1.3 is an instance of a Generalised Multiple Depot Multiple Travelling Salesman Problem (GMTSP). As discussed in Chapter 2, a wide variety of efficient and qualitative algorithms exist for the Travelling Salesman Problem (TSP). The algorithms that exist for the GMTSP are qualitatively not comparable to those of the TSP. Therefore, a transformation from a GMTSP to a TSP is proposed, so that the known algorithms for the TSP can be used. This transformation yields an asymmetric, precedence-constrained TSP. A solution to the GMTSP can then be found from a solution to the TSP.

The problem stated in Section 1.3 is formulated as a GMTSP in Section 3.1 and the transformation from a GMTSP to a TSP is described in Section 3.2. In Section 3.3 will be shown how to obtain a solution to the GMTSP from a solution to the TSP. Finally, the model needs to be able to handle single-sided connections, which are connections where multiple cables are allowed to lie in the same trench. This will be described in Section 3.4.

3.1 Formulation of the GMTSP

Let there be n targets at locations V_i for $i \in I_t = \{1, ..., n\}$; they represent the street cabinets. At most three types of cables are considered: SAN (type 0), PAN (type 1) and INN (type 2). All cables of type i are represented by one depot C_i, for $i \in I_d = \{0, ..., p\}$, $p \leq 2$. Let $V = \{V_1, ..., V_n, C_0, ..., C_p\}$ denote the set of all vertices. The set of edges E contains all existing paths between V_i and V_j, for $i, j \in I_t$, and the set of edges $\{V_1, ..., V_n\} \times \{C_0, ..., C_p\}$. An edge from street cabinet V_i to the closest cable of type j is represented by (V_i, C_j). Let there be m_i salesmen at each depot C_i, where the total number of salesmen is $m = \sum_{i=0}^{p} m_i$.

Let $c : E \longrightarrow \mathbb{R}^+$ be the cost function, where $c(u, v)$ denotes the shortest path distance of travelling from vertex u to vertex v along the road for all edges $(u, v) \in E$. Some roads may only allow one-way traffic; this is ignored meaning that the cost function is symmetric. Let $G = (V, E)$ be the corresponding undirected weighted graph.

A path $P^{(j)}$ consists of the targets and the depot that salesman j has addressed. Let C_{j_0},

$j_0 \in I_d$, denote the initial location of the salesman. If a salesman does not visit any targets, $P^{(j)}$ is an empty tour and the cost $c(P^{(j)})$ is zero. When a salesman j visits $r_j \geq 1$ distinct targets, the tour consists of $r_j + 1$ vertices and $P^{(j)} = (V_{j_1}, V_{j_2}, ..., V_{j_{r_j}}, C_{j_0})$. The initial depot of salesman j has been eliminated here, since it is necessarily the same as the final depot. The cost for such a tour is $c(P^{(j)}) = c(C_{j_0}, V_{j_1}) + \sum_{k=1}^{r_j - 1} c(V_{j_k}, V_{j_{k+1}}) + c(V_{r_j}, C_{j_0})$.

Reorder the paths $P^{(j)}$ such that the first m_0 salesmen start from depot C_0, salesmen m_0+1 to m_0+m_1 start from C_1 and the remaining salesmen start from depot C_2. Then, the tour T made by the salesmen is given by:

$$T = (C_0, P^{(1)}, ..., P^{(m_0)}, C_1, P^{(m_0+1)}, ..., P^{(m_0+m_1)}, C_2, P^{(m_0+m_1+1)}, ..., P^{(m)}).$$

The problem is formulated as follows: given the graph $G = (V, E)$, find tours $P^{(j)}$ such that each target is visited by a salesman exactly once while the total cost $\sum_{j \in \{1,...,m\}} c(P^{(j)})$ is minimised. In this problem formulation, the targets correspond to the street cabinets, the depots to the closest cable of a certain type and having m salesmen implies there will be at most m independent tours. This is an instance of the GMTSP.

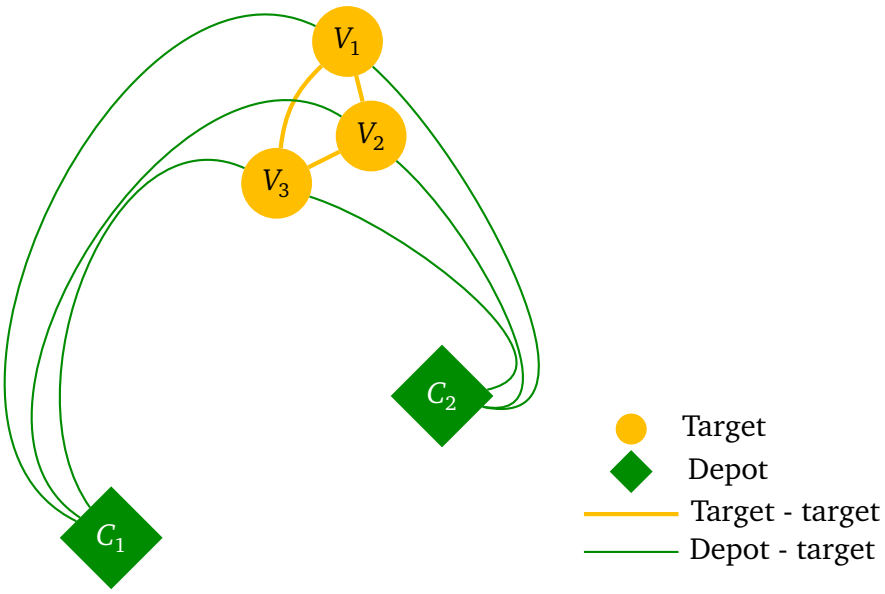


Figure 3.1 – An undirected weighted graph (weights are not shown) for three targets and two depots ($n = 3$ and $p = 2$).

Figure 3.1 shows a small graph, containing three targets, two depots and the edges contained in this GMTSP. This graph will be used as an example for the transformation described below.

3.2 Transformation from a GMTSP to a TSP

A transformation of the GMTSP to an asymmetric TSP was proposed by Oberlin et al. [16]. Based on their work, a transformation of the GMTSP defined in Section 3.1 to the precedence-constrained, asymmetric TSP is proposed in this section. The provided transformation for a graph G will result in the transformed graph G_T.

Let $C_i' = \{C_i^{(1)}, C_i^{(2)}, ..., C_i^{(m_i)}\}$ be the set of copy nodes of depot C_i for $i \in I_d$ and let $C' = C_0' \cup ... \cup C_p'$. Recall that $p \leq 2$ and that m_i is the number of salesmen at depot i. Let $V_T = V \cup C'$ be the vertex set of the transformed graph. The set of edges of the transformed graph is $E_T = E \cup \{V \times C'\}$. The transformed graph is given by $G_T = (V_T, E_T)$.

Let $c_T : E_T \rightarrow \mathbb{R}^+$ be the cost function, where $c_T(u, v)$ denotes the weight of edge (u, v) in G_T. Recall that $I_t = \{1, ..., n\}$, $I_d = \{0, ..., p\}$ and let $I_{d'}^{(i)} = \{1, ..., m_i\}$ for $i \in I_d$. The (asymmetric) cost function corresponding to G_T is defined as follows:

$$
\begin{aligned}
c_T(V_i, V_j) &= c(V_i, V_j) && \text{for all } i, j \in I_t \\
c_T(C_i, V_j) &= c(C_i, V_j) && \text{for all } i \in I_d, j \in I_t \\
c_T(C_i^{(k)}, V_j) &= c(C_i, V_j) && \text{for all } i \in I_d, k \in I_{d'}^{(i)}, j \in I_t \\
c_T(V_i, C_j^{(k)}) &= c(V_i, C_j) && \text{for all } i \in I_t, j \in I_d, k \in I_{d'}^{(j)} \\
c_T(C_i, C_i^{(1)}) &= 0 && \text{for all } i \in I_d \\
c_T(C_i^{(j)}, C_i^{(j+1)}) &= 0 && \text{for all } i \in I_d, j \in \{1, ..., m_i - 1\} \\
c_T(C_i^{(m_i)}, C_{(i+1) \mod p}) &= 0 && \text{for all } i \in I_d \\
c_T(u, v) &= \infty && \text{for all other edges } (u, v) \in E_T.
\end{aligned}
$$

The transformed graph corresponding to Figure 3.1, that contains three targets and two depots, is shown in Figure 3.2.

The intuition behind the transformed graph is explained by the example of a relay race. At the end of the race, each target must have been visited exactly once. One salesman is located at each depot and at each copy depot. All salesmen located at a depot hold a baton of a certain, unique colour. A salesman at copy depot $C_i^{(k)}$ can only accept a baton of the colour the salesman on depot C_i started with.

The starting point of a relay is a depot. The salesman at C_i starts running to the first copy depot $C_i^{(1)}$, either via some targets or directly, and hands over the baton to the salesman present there. This salesman then runs to the next copy depot $C_i^{(2)}$, again via some targets or directly. This continues until a salesman arrives at $C_i^{(m_i)}$.

If the salesman at $C_i^{(m_i)}$ decides to visit a target, he will not be able to visit C_{i+1} since the edge from a target to a depot has cost ∞. Moreover, a salesman at copy depot $C_{i+1}^{(1)}$ does not accept a baton of a salesman from $C_i^{(m_i)}$. Therefore, the only option is to go from $C_i^{(m_i)}$ directly to depot C_{i+1}. That is where the relay ends for the salesman corresponding to depot i. Note that in case $i = p$, the salesman returns to C_0 to close up the tour.

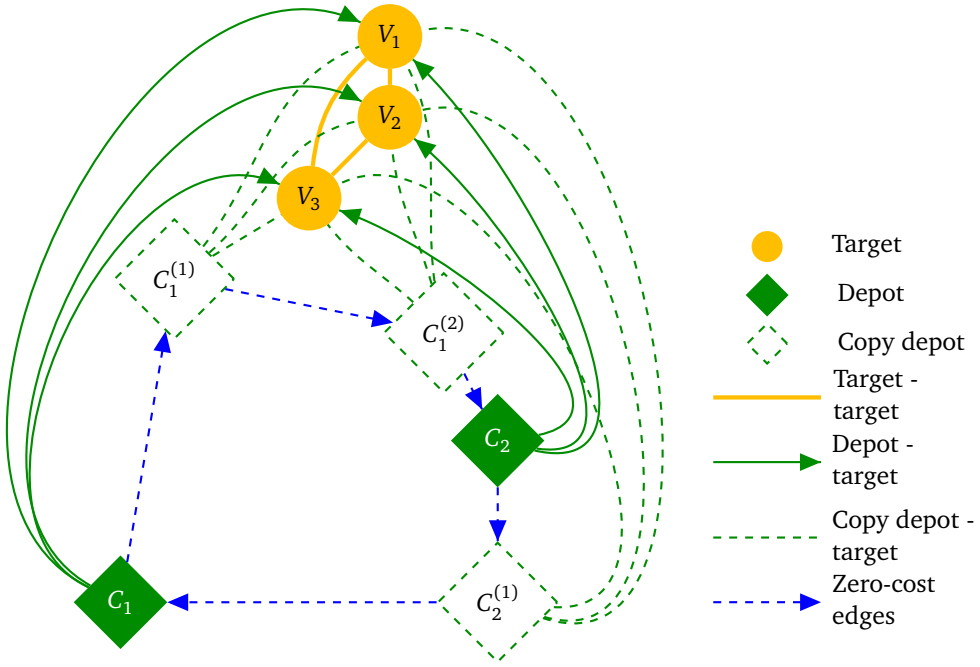


Figure 3.2 – An example containing three targets, two depots and three copy depots ($n = 3$, $p = 2$, $m_1 = 2$, $m_2 = 1$). This figure shows the corresponding transformation $G_T = (V_T, E_T)$ of which the original graph G is shown in Figure 3.1.

Let us look at these tours from a different perspective. Since each target is visited exactly once and the (copy) depots where a salesman starts and ends are fixed, the race with multiple salesmen can also be seen as one salesman who runs a large race on his own. Doing so results in a solution to a TSP, with a single salesman. For $i \in I_d$ and $j \in \{1, ..., m_i\}$, let the ordered set $P_i^{(j)} = (V_{j_1}, ..., V_{j_{r_j}}, C_i^{(j)})$ denote the path of salesmen j starting from copy depot $C_i^{(j-1)}$, visiting $r_j \geq 0$ distinct targets V_{j_k} for $k = 1, .., r_j$ and ending at copy depot $C_i^{(j)}$ (the first salesman starts from C_i). The cost of a path $P_i^{(j)}$ for $r_j \geq 1$ is given by $c_T(P_i^{(j)}) = c_T(C_i, V_{j_1}) + \sum_{k=1}^{r_j - 1} c_T(V_{j_k}, V_{j_{k+1}}) + c_T(V_{r_j}, C_i)$, otherwise the cost is zero. Then, the following tour can be constructed from the paths $P_i^{(j)}$:

$$T_T = (C_0, P_0^{(1)}, ..., P_0^{(m_0)}, C_1, P_1^{(1)}, ..., P_1^{(m_1)}, C_2, P_2^{(1)}, ..., P_2^{(m_2)}). \tag{3.1}$$

The goal is to find a tour where each target is visited exactly once, while the total cost $c_T(T_T) = \sum_{i=0}^{p} \sum_{j=1}^{m_i} c_T(P_i^{(j)})$ is minimised.

Note that this problem is an asymmetric precedence-constrained TSP, as all depots and copy depots of type i should have been visited before visiting depots and copy depots of type $i + 1$. The precedence constraints are visualised in Figure 3.3. Moreover, a depot should be visited before any of its copies: otherwise the salesman has no baton to deliver. The ordering of the copy depots of a given type is not precedence-constrained (apart from the last copy), since they can simply be renamed.

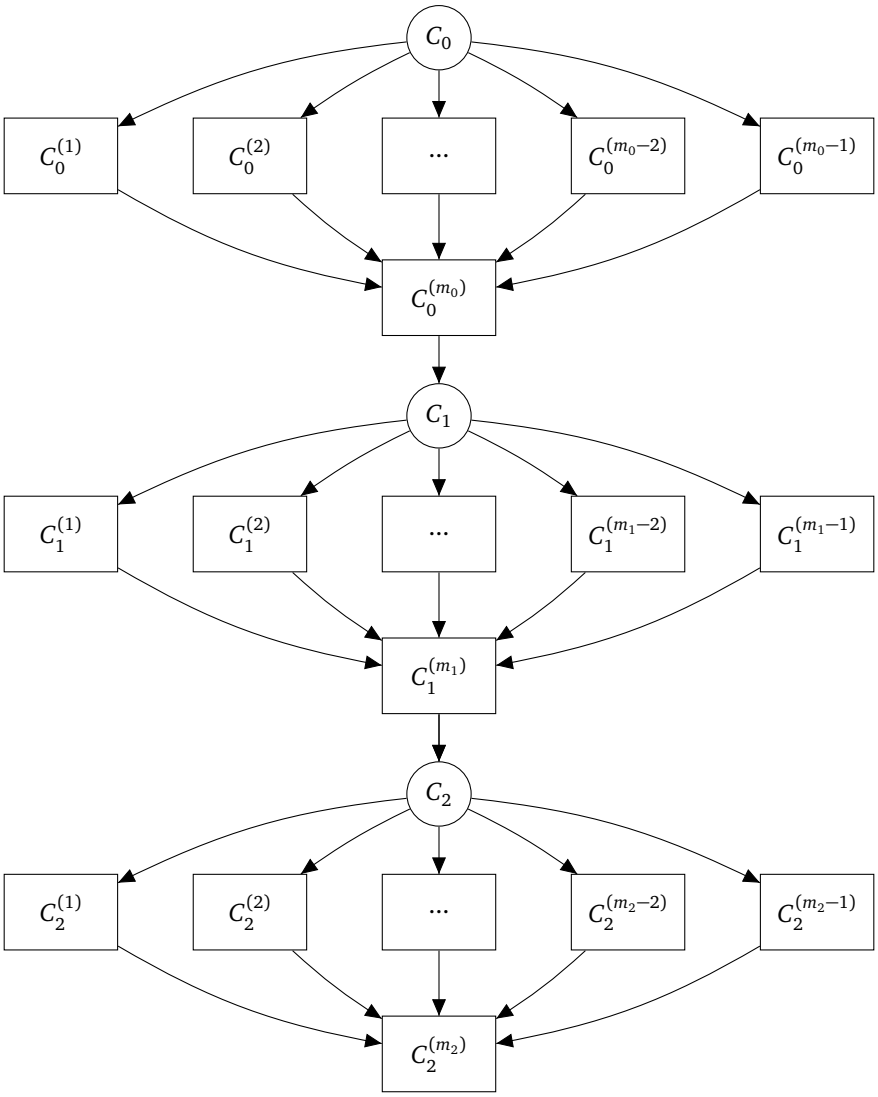


Figure 3.3 – This diagram shows the precedence constraints for the depots and copy depots. An arrow gives the ordering of the visits of the (copy) depots. All tours start with depot C_0, from which all copy depots of type 0 but the last one are reachable. Before the last copy depot of type 0 can be visited, all other copy depots of type 0 should have been visited. From $C_0^{(m_0)}$, only depot C_1 can be reached. The same holds for copy depots of type 1 and 2.

3.3 Transformation from a TSP to a GMTSP

In order to retrieve the solution to the GMTSP from T_T, consider the tours $P_i^{(j)}$ in T_T made by each salesman individually. The tour can be obtained by:

- removing all edges $(V_j, C_i^{(k)})$ and adding the edge (V_j, C_i) with cost $c(V_j, C_i)$,
- removing all edges $(C_i^{(k)}, V_j)$ and adding the edge (C_i, V_j) with cost $c(C_i, V_j)$,
- removing all zero-cost edges (apart from edges between targets and (copy) depots that happen to be of zero cost).

This will result in at most m distinct tours, where each tour ends at the depot it started from.

3.4 Single-sided connections

When at most 2000 customers are subject to street cabinets that are connected through the same cable, the cables to and from the street cabinets are allowed to lay in the same trench. Such a connection is called a single-sided or a non-redundant connection, see Figure 3.4b for an example. Tours constructed in the previous sections were redundant tours: when at least two targets are connected to the same type of cable, the cables lie in different trenches, see Figure 3.4a. Tours containing exactly one target are always non-redundant; however there are no street cabinets with more than 2000 customers in the problem set.

Costs can be reduced by allowing non-redundant connections, as the total length of the trenches that needs to be dug is smaller. However, it does increase the vulnerability of the network. If a cable breaks in a redundant tour, the data can be rerouted through the undamaged cable. On the other hand, if both cables are in the same trench, they will both be damaged and the street cabinet is disconnected from the network.

(a) A tour without single-sided connections: a redundant tour.

(b) A tour with two single-sided connections: a non-redundant tour.

Figure 3.4 – The cables and street cabinets are for illustrative purposes only, the figures do not show the actual locations of cables and street cabinets. Cables of type 0 (black) and type 1 (blue) are available, and three street cabinets (red markers) need to be activated. The red line shows two possible tours: a tour where single-sided connections are not allowed (a) and a tour where they are allowed (b). The redundant option leaves a more reliable network, the non-redundant option reduces costs.

The full tour is optimised by considering each tour $P_i^{(j)}$, as defined in Section 3.2, individually. The following assumptions are made:

- Targets within the tour are close to each other and should be in the same tour when incorporating single-sided connections.
- Targets that are not in the tour are assumed to be far away from the targets in $P_i^{(j)}$.

- Targets in $P_i^{(j)}$ are assigned to depots of type i, thus the problems reduce to problems with one type of depot.

The tours can now be treated independently.

In terms of the TSP, a single-sided connection is a path where the salesman is required to pass through some of the targets twice. This is not allowed by the definition of the TSP and therefore copy targets should be added, which yields the graph $\tilde{G}_T$. Let $V_i^{(0)}$ be the copy target for target V_i, for $i \in I_t$ and let $V' = \{V_1^{(0)}, ..., V_n^{(0)}\}$ be the set of target copies. Let $\tilde{V}_T = V_T \cup V'$ be the vertex set of $\tilde{G}_T$. The set of edges is given by $\tilde{E}_T = E_T \cup \{V_T \times V'\} \cup \{V' \times V'\}$, where V_T and E_T are the vertex and edge set of the transformed graph G_T respectively.

Up until now, the cost functions represented the physical distance between depots, targets and copy depots. Now, the cost function should depend on the path that has already been constructed since trenches may be used multiple times. The first time the trench is used, it needs to be dug open so the cost is simply the distance. The second time it is used, the trench is already open so the cost should be zero.

Let $x_{(i,j)}$ be a binary variable that indicates whether a trench is already in the constructed path: it takes value 1 if the edge (i, j) is in the constructed path and 0 otherwise. Let $f_i(V_j) = x_{(C_i,V_j)} + \sum_{r=1}^{m_i} x_{(C_i^{(r)},V_j)}$ denote the number of connections from a (copy) depot of type i to target V_j. The cost function $\tilde{c}_T$ corresponding to $\tilde{G}_T$ is then defined as follows:

$$
\begin{aligned}
\tilde{c}_T(V_i, V_i^{(0)}) &= 0 && \text{for all } i \in I_t \\
\tilde{c}_T(V_i^{(0)}, V_i) &= 0 && \text{for all } i \in I_t \\
\tilde{c}_T(V_i, V_j^{(0)}) &= c(V_i, V_j) \cdot (1 - x_{(V_i,V_j)}) && \text{for all } i \neq j \in I_t \\
\tilde{c}_T(V_i^{(0)}, V_j) &= c(V_i, V_j) \cdot (1 - x_{(V_i,V_j)}) && \text{for all } i \neq j \in I_t \\
\tilde{c}_T(V_i^{(0)}, V_j^{(0)}) &= c(V_i, V_j) \cdot (1 - x_{(V_i,V_j)}) && \text{for all } i, j \in I_t \\
\tilde{c}_T(C_i, V_j^{(0)}) &= \begin{cases} 0 & \text{if } f_i(V_j) \geq 1 \\ c(C_i, V_j) & \text{else} \end{cases} && \text{for all } i \in I_d, j \in I_t \\
\tilde{c}_T(V_j^{(0)}, C_i) &= \begin{cases} 0 & \text{if } f_i(V_j) \geq 1 \\ c(V_j, C_i) & \text{else} \end{cases} && \text{for all } i \in I_d, j \in I_t \\
\tilde{c}_T(C_i^{(j)}, V_k^{(0)}) &= \begin{cases} 0 & \text{if } f_i(V_k) \geq 1 \\ c_T(C_i^{(j)}, V_k) & \text{else} \end{cases} && \text{for all } i \in I_d, j \in I_{d'}^{(i)}, k \in I_t \\
\tilde{c}_T(V_i^{(0)}, C_j^{(k)}) &= \begin{cases} 0 & \text{if } f_i(V_k) \geq 1 \\ c_T(V_i, C_j^{(k)}) & \text{else} \end{cases} && \text{for all } i \in I_t, j \in I_d, k \in I_{d'}^{(j)} \\
\tilde{c}_T(u, v) &= c_T(u, v) && \text{for all other edges } (u, v) \in \tilde{E}_T.
\end{aligned}
$$

Consider a tour $P_i^{(j)} = (V_{j_1}, V_{j_2}, ..., V_{j_{r_j}}, C_i^{(j)})$. For simplicity, add depot C_i at the start of the tour and rename $C_i^{(j)}$ to $C_i^{(1)}$. To construct a tour $\tilde{P}_i^{(j)}$ that allows for single-sided connections, add a copy target for each target and add $\tilde{m}_j$ copy depots.

A *segment* is a part of the tour that ends with a copy depot and contains only targets and copy targets. Also, the preceding node in $\tilde{P}_i^{(j)}$ of the first target in the segment must be a (copy) depot. The copy depots allow this segment to be split into at most $\tilde{m}_i$ new segments. In a feasible tour, the target and its copy appear in the same segment. For example, a feasible tour could be

$$\tilde{P}_i^{(j)} = (C_i, V_{j_1}, V_{j_1}^{(0)}, V_{j_2}, V_{j_2}^{(0)}, ..., V_{j_{r_j}}, V_{j_{r_j}}^{(0)}, C_i^{(1)}, ..., C_i^{(\tilde{m}_j)}).$$

Note that the cost of this tour $\tilde{P}_i^{(j)}$ is equal to that of $P_i^{(j)}$: as stated above, the cost of the edge between a target and its copy is zero. Also, the cost of the edge between a copy target and another target is the same as the cost between the two targets and the same holds for a copy target and a copy depot. Finally, the cost between copy depots is zero.

The cost of a redundant tour can always be reduced when allowing single-sided connections: removing any edge between two targets in the tour yields two segments that each form a single-sided connection. Recall that Figure 3.4a shows an example of a redundant tour. When an edge between a (copy) depot and a target is removed (for example as is done in Figure 3.4b), the resulting tour consists of one segment that is connected through a single-sided connection. A more elaborate discussion is given in Section 6.4.

The full tour that allows single-sided connections, $\tilde{T}_T$, can now be constructed. First, remove all copy depots from $\tilde{P}_i^{(j)}$ that do not connect targets. Then, remove the first node, which is the depot. Now, the tour is given by:

$$\tilde{T}_T = (C_0, \tilde{P}_0^{(1)}, ..., \tilde{P}_0^{(m_0)}, C_1, \tilde{P}_1^{(1)}, ..., \tilde{P}_1^{(m_1)}, C_2, \tilde{P}_2^{(1)}, ..., \tilde{P}_2^{(m_2)}).$$

Finally, rename the copy depots in $\tilde{T}_T$ such that they are ordered.

Chapter 4

Heuristic model for redundant tours

The model for solving the precedence-constrained TSP is based on work by Lin and Kernighan [12], who proposed an effective heuristic algorithm for a standard symmetric Travelling Salesman Problem. The method roughly consists of the following steps:

1. Construct a feasible initial tour T'.
2. Attempt to find an improved feasible tour T from T'.
3. If tour T is shorter than tour T', repeat step 2 starting from tour T.
4. If no improvement can be found, restart from step 1. Quit when results are satisfactory, or when computation time runs out.

Several procedures to construct an initial tour are proposed in Section 4.1, which are used in step 1 of the above outline. In Section 4.2, a closer look is taken at the second step, to improve the initial tour. Step 2 of the original method has been altered in such a way that it's suitable for a small, asymmetric and precedence-constrained TSP. Step 2 denotes one *iteration* of the algorithm. This iteration is performed multiple times, as it says in step 3, according to a suitable stopping criterion that is descibed in Section 4.3. In Section 4.4 is described how the tour found in step 4 is used as a new feasible, initial tour in step 1. This procedure is used to move to a global optimum in case a local one is found.

4.1 Tour construction heuristics

Tour construction methods are used to obtain an initial path through all targets and (copy) depots, which corresponds to step 1 in the outline of the model above.

Lin and Kernighan [9] were convinced that their algorithm would find the same local optima when starting from a random initial tour as when starting from a tour found by a tour construction procedure. Helsgaun [9] found that the quality of the final solution does not necessarily depend on the initial tour chosen, however the computation time can be reduced by choosing an initial tour close to optimal. Therefore, tour construction procedures that have proven to give accurate solutions to small TSP problems in the past, and methods that do not necessarily yield promising results are discussed.

In [6], Golden et al. compare and discuss several tour construction procedures for relatively small problems, ranging from 25 to 150 nodes. They found that the farthest insertion and arbitrary insertion methods gave suprisingly good results. Moreover, the nearest neighbour heuristic is a method that is known to perform poorly in general, which makes it an interesting heuristic to test the quality of the local search algorithm.

4.1.1 Insertion heuristics

Insertion heuristics start from an initial node or path. Nodes are inserted until a Hamiltonian cycle is found. The precedence constraints (recall Figure 3.3) impose a unique ordering on the depots and copy depots so start from the following path:

$$(C_0, C_0^{(1)}, .., C_0^{(m_0)}, C_1, C_1^{(1)}, ..., C_1^{(m_1)}, C_2, C_2^{(1)}, ..., C_2^{(m_2)}).$$

Two tour construction procedures that are considered are the arbitrary insertion (AI) heuristic and the Farthest Insertion (FI) heuristic.

For the AI heuristic, arbitrarily select a node k that is not yet in the tour. Find which edge (i, j) minimises the costs $c(i,k)+c(k,j)-c(i,j)$ and insert node k between i and j. Repeat this until all nodes are in the path.

For the FI heuristic, select a node k that is not yet in the tour, furthest away from any node in the tour and find the edge (i, j) that minimises the costs $c(i,k)+c(k,j)-c(i,j)$. Then, insert node k between i and j and repeat this until all nodes are in the path.

Both algorithms require $\mathcal{O}(n^2)$ operations since finding the appropriate edge (i, j) requires $\mathcal{O}(n)$ operations. The worst case behaviour, the ratio of the length of the arbitrary insertion tour to the length of the optimal tour, is known to be at most $2\ln(n) + 0.16$, where n is the number of nodes in the path [6].

4.1.2 Nearest neighbour heuristic

For the NN algorithm, select an initial node that is not yet in the tour and continuously add the node that is closest to the last node added and not yet in the path. In order to obtain a feasible path, start from depot C_0. Then, all (copy) depots immediately follow as they can be connected through zero-cost edges. To obtain a feasible path, all targets must be added before the last copy depot, so that the tour is closed by $C_2^{(m_2)}$.

The algorithm requires $\mathcal{O}(n^2)$ operations and the worst case behaviour, the ratio of the length of the nearest neighbour tour to the length of the optimal tour, is known to be at most $\frac{1}{2}\log_2(n) + \frac{1}{2}$, where n is the number of nodes in the path [6].

4.2 Improve the current tour

In each iteration of the algorithm, the tour is improved by performing an r-OPT move. An r-OPT move attempts to obtain a better tour by replacing r edges that are currently in the tour by r new edges, which again yields a feasible and connected

tour. These edges are picked by iteratively constructing edge sets $X = \{x_1, ..., x_r\}$ and $Y = \{y_1, ..., y_r\}$ so that when all edges in X are removed and replaced by the edges in Y, a feasible tour of lower cost is obtained. In other words, the *gain*, the difference between the cost of the old and new tour, must be positive.

However, many choices for X and Y do not yield feasible tours: the precedence constraints must be taken into account. Recall from Chapter 3, Figure 3.3, that the depot and copy depots of type i should be visited before those of type $i+1$. Also, a depot should always be visited before a copy depot of the same type.

The number of targets in our problems is small, however the number of (copy) depots compared to the total number of nodes is large. Due to the precedence constraints, this will leave very few valid exchanges for $r \geq 4$. Therefore, r-OPT moves for $r \geq 4$ in step 2 of the algorithm are not taken into consideration. Also, many 4- and 5-OPT moves can be obtained by sequentially performing 2- and 3-OPT moves.

An iteration takes the current tour and an initial node as input, from which the first edge to be removed, x_1, is determined. To keep the tour connected after performing the move, edges x_i and y_i should share an endpoint, just as y_i and x_{i+1} (see Figure 4.1). We can then write $x_i = (t_{2i-1}, t_{2i})$ and $y_i = (t_{2i}, t_{2i+1})$ (and $y_r = (t_{2r}, t_1)$) for nodes t_i in the tour ($i \leq r$).

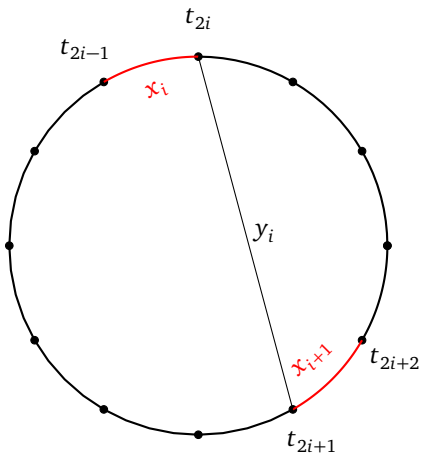


Figure 4.1 – To keep the tour connected, edges x_i and y_i must share and endpoint, just as y_i and x_{i+1}.

The gain criterion is discussed in Section 4.2.1. Then, the conditions that the edges x_i and y_i must satisfy in order to enter the sets X and Y are discussed in Section 4.2.2 and Section 4.2.3 respectively.

4.2.1 Gain criterion

Edges are only allowed to enter the sets X and Y if the difference in cost between the old and the new tour, the gain, is positive. Let $g_j = |x_j| - |y_j|$, where $|x_j|$ and $|y_j|$ are the costs of edges x_j and y_j respectively. Then, the gain of removing and adding i edges that are in the sets X and Y respectively is given by $G_i = \sum_{j=1}^{i} g_j$. To see that G_i is also

the difference between the length of the old and the new tour, let $c(T')$ denote the cost of the old tour and $c(T)$ the cost of the new tour. Then, T' and T both contain all edges that are not in X, nor in Y. Therefore, $c(T')-c(T) = \sum_{j=1}^{i} x_j - \sum_{j=1}^{i} y_j = \sum_{j=1}^{i} g_j = G_i$. Since only improvements of the tour are allowed, the edges x_i and y_i are only allowed to be added to X and Y if G_i is strictly positive. Note that this does not put a restriction on g_i directly: g_i may be negative, as long as G_i is strictly positive.

It may seem that this criterion is too restrictive. However, Lin and Kernighan [12] state that if a sequence of numbers has a positive sum, there is a cyclic permutation such that every partial sum G_i is positive. The sequence of numbers g_k can be arranged in descending order. Then, the partial sum G_i is the sum of the first i terms in this sequence. As long as the sum of the sequence is positive, so is G_k for $k \leq i$ since the terms g_k are ordered descendingly. In other words: whenever $G_i < 0$ is found for the first time for some i, the edges x_i and y_i should not be added to X and Y but the procedure for the current X and Y should be performed (when they contain more than one edge). That will yield the largest gain, given the edges currently in X and Y.

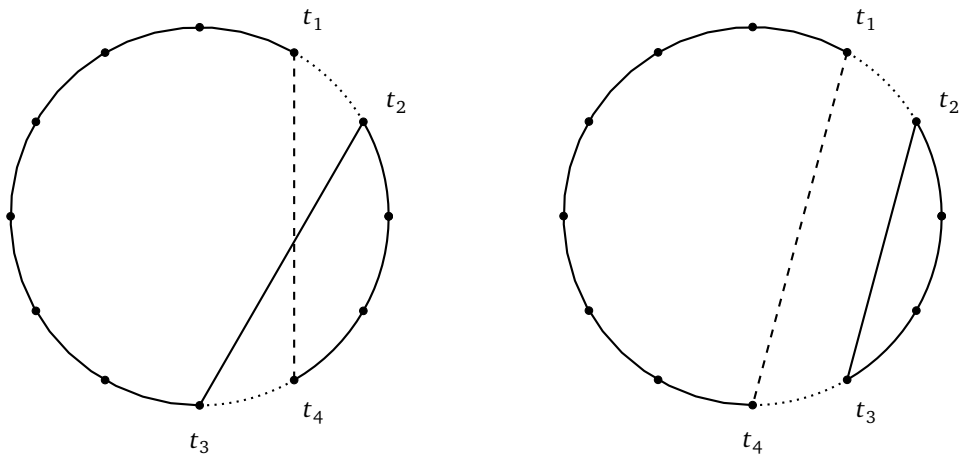
The edges that enter the sets X and Y are determined iteratively. When $G_i < 0$ or when the sets contain three edges (since only 2- and 3-OPT moves are performed), no more edges are added to the sets. These sets X and Y are always closed tours (for sets with at least two edges). A move with only one edge in each set is not valid, so ignore these sets and try a new iteration with a new starting node. In case there are two edges in each set, the 2-OPT move can be performed. However, when both sets contain three edges, the 2-OPT move may yield a larger gain than the 3-OPT move. The corresponding 2-OPT move removes the edges x_1 and x_2 from the tour, and adds the edges y_1 and the edge that closes up the tour (which is not y_2). The move that yields the largest gain is performed.

4.2.2 Choices for edges in X

The algorithm takes the current tour and one randomly picked node as input. The edge x_1 is picked in such a way that the input node is the endpoint on the right. The second endpoint is found by moving in an anti-clockwise manner. However, if this does not yield a valid choice for x_1, the input node will be the left endpoint of x_1. If both choices for x_1 are valid, but the chosen edge does not yield a positive gain, do not consider the other option for x_1: picking another node as starting point will yield this edge.

The edges $(C_i^{(m_i)}, C_{(i+1) \bmod p})$ (where p is the number of the largest type of depot, $p \leq 2$) cannot be broken. There is no other valid link to $C_{(i+1) \bmod p}$ than $C_i^{(m_i)}$ (recall the relay race in Chapter 3). When such an edge is broken, the largest gain is obtained by adding an invalid edge, since that is when $g_i = 0$. Obviously, this is undesirable, so breaking of the previously mentioned edges is not allowed.

Finally, when both sets contain three edges but the gain is largest when performing a 2-OPT move with x_1, x_2, y_1 and the edge that closes the tour, this must yield a feasible tour. Therefore, edges $x_i = (t_{2i-1}, t_{2i})$ and $y_i = (t_{2i}, t_{2i+1})$ may only be added to X and Y respectively if edge x_i and edge (t_{2i}, t_1) close up a tour for $i \geq 2$. Figure 4.2 shows



(a) A valid choice for x_2, since closing up the tour by connecting t_1 and t_4 yields a feasible tour.

(b) An invalid choice for x_2, since closing up the tour by connecting t_1 and t_4 does not yield a feasible tour.

Figure 4.2 – A valid and an invalid choice for $x_2 = (t_3, t_4)$, given $x_1 = (t_1, t_2)$ and $y_1 = (t_2, t_3)$.

a valid and an invalid choice for $i = 2$.

4.2.3 Choices for edges in Y

When removing the edges in X from the tour when performing an r-OPT move, the tour is divided into r segments which are referred to as s_1, s_2, ..., s_r. Without loss of generality, assume that depot C_0 is in segment s_1, and that segment s_i is passed through before s_{i+1} for $i \leq r-1$ when the tour is traversed in a anti-clockwise order.

The sets X and Y must be disjoint, meaning that edges that have been removed cannot be added in the same iteration and vice versa. Moreover, the precedence constraints must be taken into account, which strongly reduces the number of edges that can be added to the set Y. Below, we discuss valid choices for these edges for 2- and 3-OPT moves.

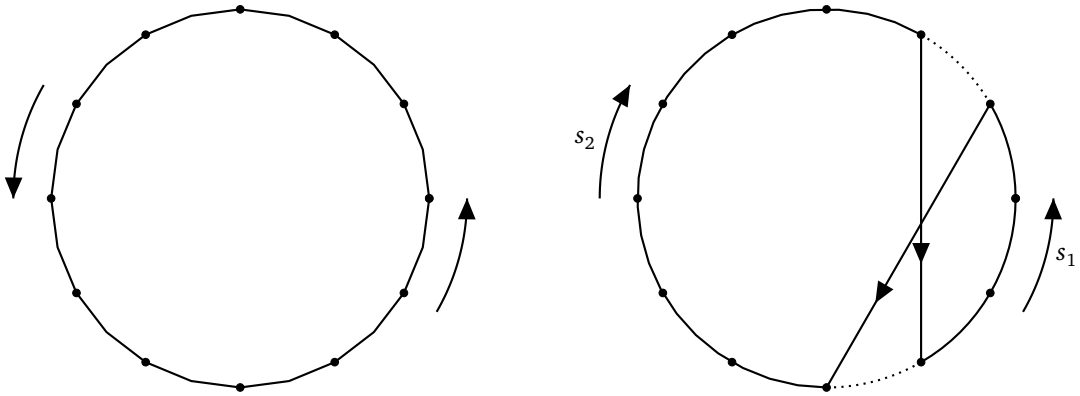
Precedence-constrained 2-OPT procedure

Figure 4.3 shows a 2-OPT move: two edges are removed from the tour and two new ones are added. A 2-OPT move will always reverse the direction of s_2, which means the resulting tour is only feasible when s_2 contains at most one type of copy depot and possibly the corresponding depot. Otherwise, the precedence constraints are violated.

Precedence-constrained 3-OPT procedure

Figure 4.4 shows all possible 3-OPT moves including the directions of the segments. Edges x_1, x_2 and x_3 divide the original tour in three segments: s_1, s_2 and s_3.

Figure 4.4a, Figure 4.4b and Figure 4.4c show a tour in which s_3 is traversed before s_2. Therefore, it must be checked whether s_2 contains (copy) depots of a type preceding



(a) Original tour. The path is traversed in an anti-clockwise manner.

(b) A possible 2-OPT move. The two edges that have been removed split the tour in two segments: s_1 and s_2.

Figure 4.3 – Check the feasibility of the 2-OPT move on the original tour. Whether the move is feasible depends on the nodes in s_2, since the direction of this segment is reversed.

(copy) depots in s_3 since that would make the tour infeasible. Therefore, a feasible tour is constructed if:

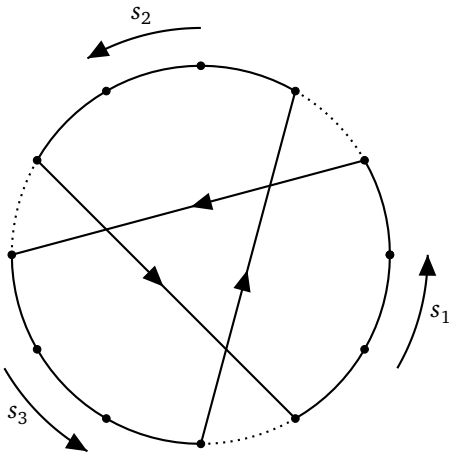
- s_2 and s_3 do not contain any (copy) depots,
- s_2 contains exactly one type of (copy) depot and s_3 does not contain any (copy) depot, or vice versa,
- s_2 contains exactly two types of (copy) depots and s_3 does not contain any (copy) depot, or vice versa,
- s_2 contains exactly three types of (copy) depots and s_2 does not contain any (copy) depot, or vice versa,
- s_2 and s_3 both contain exactly one type of (copy) depot, which is the same.

Moreover, Figure 4.4b and Figure 4.4c show that nodes in segments s_3 and s_2 are reversed respectively, which means an infeasible tour is found whenever the reversed segment contains two or more types of (copy) depots.

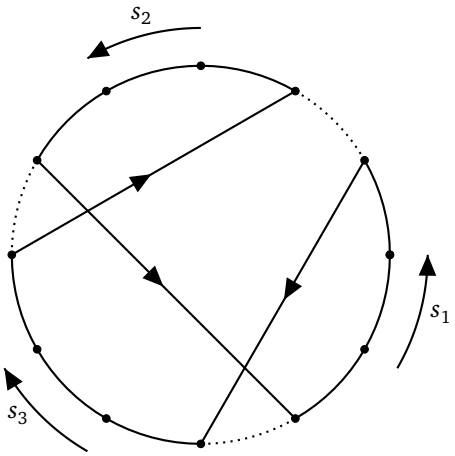
Finally, Figure 4.4d shows that the the order of the segments is preserved, but the nodes within s_2 and s_3 are reversed. Therefore, both s_2 and s_3 can contain at most one type of (copy) depot and no other restrictions apply.

4.3 Stopping criterion

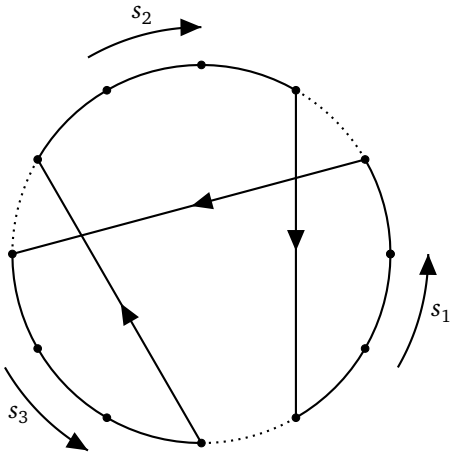
In Section 4.2, one iteration of the model is described, which corresponds to the second step of the general outline that was given in the beginning of this chapter. After multiple iterations have been performed (step 3), the procedure must be restarted if no improvement can be found. In other words, a stopping criterion is needed that decides whether an improvement is probably not found anymore.



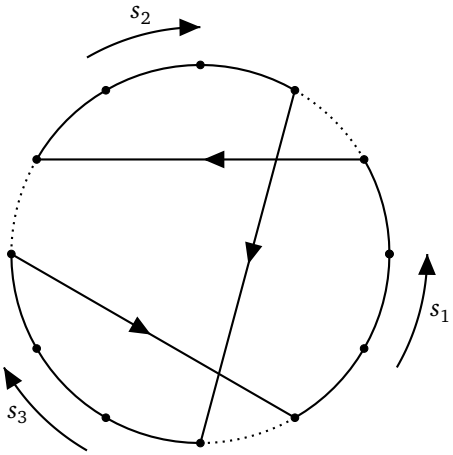
(a) The order of the nodes is preserved within each segment.



(b) The ordering is preserved within s_1 and s_2, but not in s_3.



(c) The ordering is preserved within s_1 and s_3, but not in s_2.



(d) The ordering is preserved within s_1, but not in s_2 and s_3.

Figure 4.4 – All possible 3-OPT moves for a precedence-constrained graph. The removed edges split the tours in segments s_1, s_2 and s_3.

Let n denote the number of nodes in the tour. Then, we consider it very unlikely that an improvement is found after having picked n random starting nodes, of which none resulted in an improved tour. Since the number of nodes n is small for the problems that are considered, this will result in acceptable running times. More restrictive stopping rules may be necessary when larger problems must be solved.

4.4 Kick a tour

After performing step 2 and 3 of the algorithm, a tour T that is shorter than the initial tour T' is hoped to be found. At least, the tour cannot be of larger cost than T' since only positive gains were accepted during the iteration. However, we may reside in a local optimum that is not the global one.

To overcome this problem, Martin et al. [15] propose to *kick*, i.e. perturb, the tour in such a way that it is hard to undo by the heuristic algorithm. Otherwise, the algorithm would simply lead back to the original tour without gain. Since 2- and 3-OPT moves are performed in an iteration, it would be natural to perform a 4-OPT move as a kick. However, some 4-OPT moves can be performed by iteratively performing 2- and 3-OPT moves which would lead straight back to the original tour. Preferrably, the perturbation is also succesful when the heuristic is extended to perform 4-OPT moves, in case the algorithm is used for solving for bigger problems. However, as Lin and Kernighan [12] mentioned in their paper, only feasible moves can be performed during an iteration, which means the tour is feasible in every step of the iteration, since edges are added iteratively X and Y. Therefore, a non-sequential move will suit both goals.

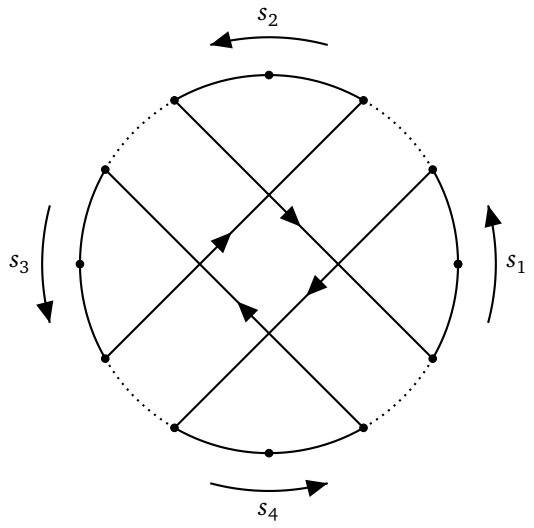


Figure 4.5 – A non-sequential 4-OPT move, also referred to as the double-bridge move.

Consider a non-sequential 4-OPT move to kick the tour. The move is shown in Figure 4.5 and is also referred to as a double-bridge move, introduced by Martin et al. [15]. An increase in cost of at most 20% is accepted by this move, which hopefully enables the heuristic to find the global optimum. The construction of the double-bridge move is described below.

There are several ways to implement the double bridge move. First, determine the edges to be removed by randomly picking one endpoint of each edge and traversing the graph in an anti-clockwise order to obtain the second endpoint of each edge. By removing these edges from the path four segments are obtained, $s1$ through s_4, as shown in Figure 4.5.

The move is performed by reordering the segments so that they are traversed in the order s_1, s_4, s_3, s_2. The order of the nodes within the segments is preserved. Note that the edges y_1 to y_4 are not explicitly obtained. Recall the precedence constraints from Figure 3.3. A feasible tour is constructed if:

- s_3 does not contain (copy) depots preceding (copy) depots in s_4, and

- s_2 does not contain (copy) depots preceding (copy) depots in s_3.

Moreover, the edges $(C_i^{(m_i)}, C_{(i+1) \bmod p})$ for $i \leq p$ cannot be broken as that would result in an infeasible tour. The above procedure is repeated until a valid double bridge move is found that gives an increase in cost of at most 20%. However, it might occur that the problem is small and that the current tour does not allow for a feasible double bridge move. In that case, an initial tour is constructed by the random insertion heuristic, as described in Section 4.1.

Chapter 5

Branch and bound method for redundant tours

The heuristic method described in Chapter 4 does not give any guarantees on the quality of its solutions. Therefore, an exact branch and bound (B&B) algorithm is used to determine the quality of these tours. A B&B algorithm divides the set of all feasible solutions into subsets and then evaluates them systematically, until the optimal solution is found.

An exact method needs to evaluate all feasible permutations and thus is usually time consuming, more so than an efficient heuristic method. Since the problem sizes will be scaled up from MA to MC areas, recall Section 1.3, the chosen method (either heuristic or exact) must be able to handle problem sizes larger than the ones discussed in this thesis. Therefore, the B&B method is used as a measure for the quality of the heuristic method. With this method, upper and lower bounds on tours are derived which give insights on the tour quality. These bounds can be used for problem sizes that are too large to handle for an exact B&B algorithm.

Recall from (3.1) in Section 3.2 that a tour consists of depots and multiple segments, where a segment contains one or more targets and ends with a copy depot. Since a tour is closed, either of the nodes can serve as a starting point. An intuitive choice is to start from the first depot, C_0. All possible paths then emerge as branches in a tree from this root node.

To find the optimal tour, the tree is traversed by applying a depth-first search. Starting from the root of the tree, a branch is explored as far down as possible. By moving up to the last expanded node, the next branch can be explored. See Figure 5.1 for the order through which nodes in a tree are passed for a problem of 5 nodes.

The tree contains many equivalent, or duplicate paths. For example, the paths $(C_0, V_1, V_2, C_0^{(1)}, V_3, C_0^{(2)})$ and $(C_0, V_3, C_0^{(1)}, V_1, V_2, C_0^{(2)})$ are equivalent. Traversing all duplicate paths is highly inefficient and therefore detection of a duplicate is needed as high up in the tree as possible. Section 5.1 describes how to detect a path that has been traversed before.

Moreover, another type of duplicates appear in the tree. The number of permuta-

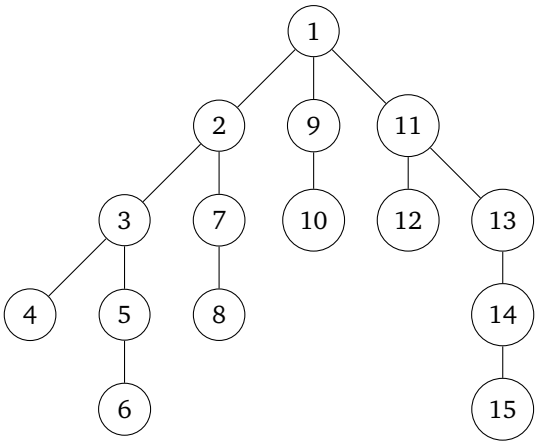


Figure 5.1 – Traversing a graph for a problem containing 5 nodes in a depth-first order. The node numbering corresponds to the order through which the nodes are traversed.

tions of a tour of n nodes with m (copy) depots (where copy depot $C_i^{(j)}$ is visited before $C_i^{(j+1)}$) is $\mathcal{O}((n-m)!)$: for each location in the tour, there are $n-m$ possible targets and there is one possible (copy) depot, since they are precedence constrained. Therefore an exhaustive search is not an option, even for small n. If it were, the use of a heuristic method wouldn't be necessary in the first place. Therefore, a rule is needed that determines whether the branch that is currently explored does definitely not contain the optimal solution. In that case, the next branch should be explored and the current node is *pruned*, i.e. not explored any further. The rules are based on upper and lower bounds on the tour cost and they will be discussed in Section 5.2.

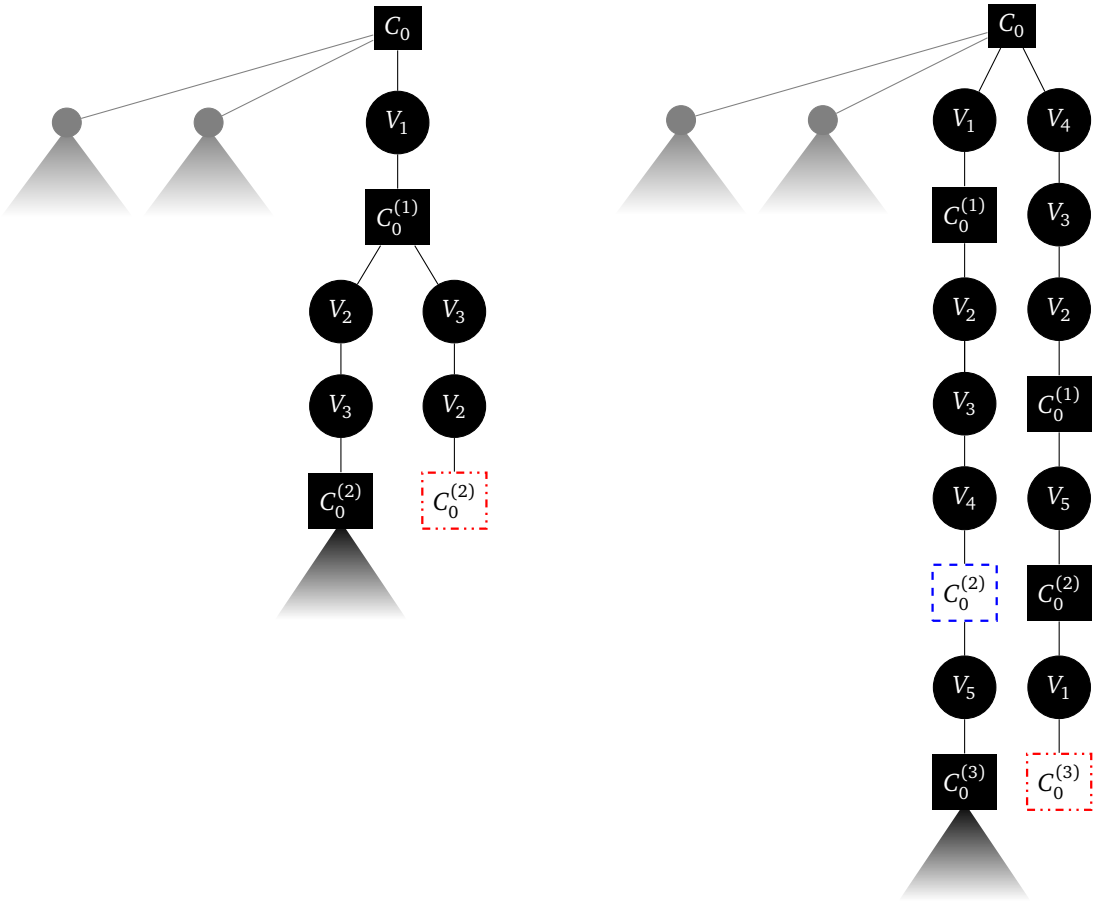
5.1 Duplicate elimination

In the heuristic model of Chapter 4, the order in which copy depots of the same type occurred in a tour was not important, as the efficiency of the program was not influenced by it and the copies could simply be renumbered to create a feasible tour. However, allowing any ordering of the copy depots of the same type in the B&B method leaves many duplicates in the tree, which unnecessarily decreases the efficiency of the method. Recall that p denotes the number of the largest depot type and that m_i denotes the number of copy depots of type i. Therefore, we require that copy depot $C_i^{(j)}$ is visited before copy depot $C_i^{(j+1)}$. Also, C_{i+1} must be visited immediately after $C_i^{(m_i)}$ for $i < p$: this condition makes sure a salesman is allowed to move from one type of depot to another type (recall the relay race in Section 3.2). The tour is finished after the last copy depot of type p is visited and all targets have been passed.

The tree contains many duplicate, i.e. equivalent, branches. For example, consider the tour $T = (C_0, V_1, V_2, V_3, C_0^{(1)}, V_4, C_0^{(2)}, V_5, V_6, C_0^{(3)})$, that contains six targets and three copy depots. This tour contains three segments $P_i^{(j)}$, where i denotes the type of copy depots: $P_0^{(1)} = (V_1, V_2, V_3, C_0^{(1)})$, $P_0^{(2)} = (V_4, C_0^{(2)})$ and $P_0^{(3)} = (V_5, V_6, C_0^{(3)})$.

Let us consider the duplicates of T. First, the segments that contain at least two targets, i.e. $P_0^{(1)}$ and $P_0^{(3)}$, can appear in two ways: reversing the targets in a segment does not change the tour, which gives four equivalent representations of the given

path T. Moreover, the order in which the three segments appear does not affect the tour, which results in $3! = 6$ equivalent tours. Therefore, equivalences of T occur $2^2 \times 3! = 24$ times in the tree. Note that the location of the copy depots is fixed when the location of the segments is known, as well as the ordering of the depots and copy depots. The number of duplicates of a specified path containing one type of depot is given by $s_i! \cdot 2^{s_{i,2+}}$, where $s_{i,2+}$ denotes the number of segments with at least two targets and s_i denotes the total number of segments subject to (copy) depots of type i. For paths containing $p+1$ types of depots, the number of duplicate paths is given by $\Pi_{i=0}^{p} s_i! \cdot 2^{s_{i,2+}}$.



(a) The reverse of (V_3, V_2) already appears in a branch occurring from $C_0^{(1)}$ which means that the current node $C_0^{(2)}$ must be pruned. Otherwise, the branches occurring from the current node $C_0^{(2)}$ will yield the same branches as the ones occurring from $C_0^{(2)}$ in the left path.

(b) The three segments already appear in another path occurring from the root of the tree in a different ordering. Therefore, the current node $C_0^{(3)}$ must be pruned, since it's a duplicate of a path that's already in the tree.

Figure 5.2 – Two scenarios for duplicate elimination. On the left: check for segments from the last expanded (copy) depot that have already been added before in a reversed order. On the right: the current path already occurs in a different ordering of the segments, possibly reversed, in the tree. The last added node (red dash-dotted) is pruned. This node is also pruned if a previous equivalent path has been pruned, for example at the blue dashed copy depot.

The number of duplicates grows rapidly as the number of segments grows. To increase the efficiency of the algorithm, a duplicate should be detected as high up in the tree as possible. The detection can only occur when the next node to add is a copy depot, so that all segments that are currently in the path are complete. Figure 5.2 shows two cases that are discussed below.

- Consider the last expanded (copy) depot and let the current segment contain at least two targets. Then, when the reverse of the current segment has already been added to the last expanded (copy) depot, the current path is a duplicate so the last node is pruned. In Figure 5.2a, the last expanded copy depot is $C_0^{(1)}$. The reverse of (V_3, V_2) has already been added to $C_0^{(1)}$, so the current copy depot $C_0^{(2)}$ is pruned.

- Consider the path traversed from C_0 down to the last added copy depot. To check whether the path on the right is already in the tree in a different order, all paths that have previously been traversed must be checked (also the pruned ones). See Figure 5.2b for an example of two duplicate paths. Consider all segments in each path up to node $C_0^{(3)}$ and check whether the current path is a duplicate by looking for both identical and reversed segments. If equivalent segments for all segments in the current path occur, the node can be pruned. Note that it is not possible to prune node $C_0^{(2)}$ in the path on the right already: to be sure only duplicate paths are removed, check the segments in each path on the same level in the tree.

  In addition, sometimes nodes can be pruned even though not all segments in the current path occur in a previously passed path. This is the case when paths were pruned before all targets in the segments were visited. For example, consider the path on the left in Figure 5.2b again, that has been pruned at $C_0^{(2)}$ (the node with a blue, dashed border). Then, the last node in the path on the right can also be pruned, since all segments in the previously visited path occur in the current path and all branches emerging from this node were not worth visiting.

The goal of detecting a duplicate path is to increase the efficiency of the B&B method, which is done by the measures discussed above. Another way to do so is by determining upper and lower bounds to a tour, both up front and during tree traversal. These bounds detect branches that do definitely not contain the optimal solution. Suitable bounds are discussed in the next section.

5.2 Upper and lower bounds

Consider a path that is traversed up to node v_{p-1} at level $p-1$. Let v_i, $1 \leq i \leq p-2$ be the nodes previously visited in this path, where the index corresponds to the level in the tree. These nodes can either be targets, depots or copy depots. See Figure 5.3 for an illustration of a tree where some paths have been traversed, and the last path has been traversed until node v_{p-1}. Node v_p is currently explored. A rule is needed that detects whether the optimal solution is definitely not in the branch that will originate from v_p. Then, either v_p is added, or v_p is pruned.

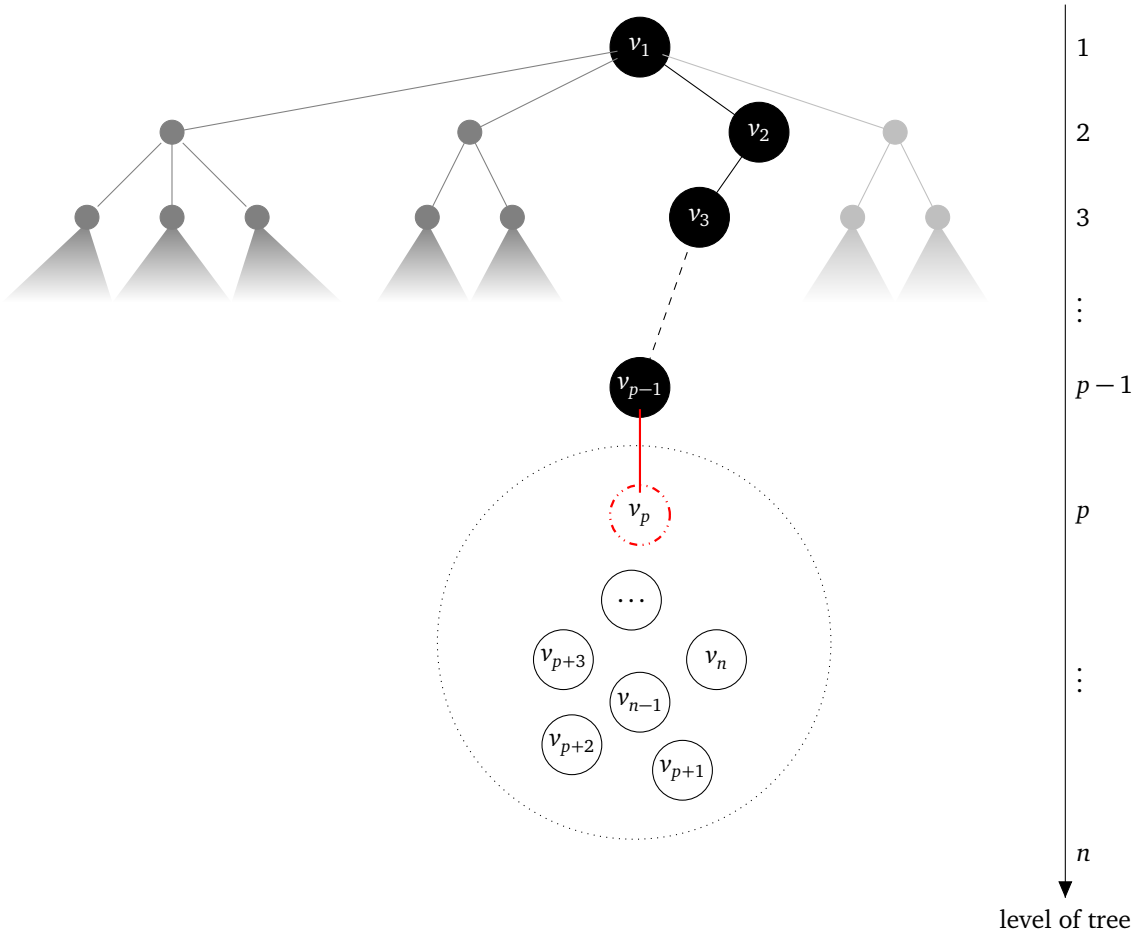


Figure 5.3 – Consider the extension of the path v_1 to v_{p-1} by node v_p. To decide whether to prune or to continue branching from node v_p, determine a lower bound on the cost of the tour, given v_1 to v_p. Do this by constructing a minimum spanning tree through the remaining nodes and adding the cost of the minimum spanning tree to the cost of the tour v_1 to v_p. If the resulting lower bound exceeds the upper bound, the optimal tour cannot appear in this branch so prune v_p. Otherwise, continue branching by adding v_p to the current tour.

First, an upper bound on the cost of the tour can detect a branch that does not contain an optimal solution. If the cost of the path traversed from v_1 to v_p exceeds this upper bound, the optimal tour is not in the branches emerging from v_p and v_p can be pruned. The cost of the full tour found by the heuristic method of Chapter 4 serves as an upper bound on the optimal solution.

During tree traversal, the upper bound is adjusted if the current path contains all nodes and is of lower cost than the upper bound. This is possible because the tree is traversed in a depth-first order (see Figure 5.1). Moreover, when exploring the children of the current node, i.e. the nodes one level deeper, the visits are ordered by the cost of the connecting edge so that the closest node is visited first. In this way, if a bad upper bound is used, a better one is quickly found. Also, chances of finding the optimal tour in an early stage of the tree traversal increase. However, pruning a node based solely on an upper bound is inefficient, as this usually means many nodes have already been traversed.

A lower bound can be used to detect a branch that does not contain the optimal solution. A lower bound is a cost that is at least required to make a complete tour. It may not be possible to find a feasible tour at this cost, but the cost of the optimal tour is never smaller than this lower bound. The goal is to find a strong lower bound, which means that this bound should be close to the cost of the optimal tour.

This bound is determined during traversal of the tree. At level p, there are $n-p$ nodes left that still need to be traversed. A lower bound on the cost of the total tour can be found by determining a minimum cost of traversing through these $n-p$ nodes. Then, adding this cost to the cost of the path from v_1 to v_{p-1} and adding the least cost edge from v_p to any node in the set of non-traversed nodes gives a lower bound on the tour cost, given the path from v_1 to v_p. When this lower bound exceeds the upper bound described above, the branches emerging from v_p do not contain optimal tours and consequently this node can be pruned.

In the early stages of traversing the tree, this lower bound appears to be weak since many edges between the remaining nodes are of cost zero. Therefore, a stronger lower bound on the full tour is determined up front. During the traversal of the tree, the maximum of these lower bounds is used. Section 5.2.1 describes how to find a lower bound on the tours in the current branch at level p and Section 5.2.2 describes how to obtain a bound on the full tour, both based on a minimum spanning tree. Section 5.2.3 gives a method to determine a lower bound through all nodes before traversing the path based on the fact that a node in a tour must have degree two.

5.2.1 Lower bound during tree traversal

A lower bound on the cost of a certain tour can be obtained by constructing a minimum spanning tree (MST) through its nodes. A tree is a directed acyclic graph with n nodes and $n-1$ edges. A tour through n nodes also contains $n-1$ edges. A tour is a special case of a tree, since the starting point and endpoint of a tour are not connected. Therefore, the cost of an MST starting at level p gives a lower bound on the cost of the tour through the remaining $n-p$ nodes.

A minimum spanning tree is constructed by Kruskal's algorithm [11], which works as follows:

- Start from the edge with lowest cost.
- Continuously add the edge of lowest cost that does not create a cycle.
- Repeat until all nodes are in the tree.

Allowing cycles in the MST might lead to multiple disconnected trees, but the tour should be connected. Therefore, creating cycles is not allowed.

To obtain a lower bound on the optimal tour in the current branch, a MST is constructed through the nodes $v_{p+1}, ..., v_n$ that have not yet been traversed. Then, a lower bound is given by the cost of an MST plus the cost of the tour from v_1 to v_p and the least cost edge from v_p to the MST.

5.2.2 Lower bound on a complete tour (MST)

The bound discussed in the previous section can also be used to determine a bound on the full tour. An MST is then constructed by Kruskal's algorithm through all nodes in the problem.

The gap between the cost of the optimal tour and the cost of an MST through a set of nodes is at most 50%, as is made plausible below. Instead of using an MST as a lower bound, an MST can also be used to obtain an upper bound on the tour cost, for example as stated in [14]. This is done by doubling the edges in the tree, so that all edges are traversed exactly twice. By shortcutting edges, i.e. removing all edges so that costs are reduced while the tour remains feasible, a closed and feasible tour is obtained. Then, as the cost of at most twice this MST serves as an upper bound, the lower bound obtained by the MST is at least half of the cost of the optimal tour.

An improvement to this lower bound is made by noting that one of the zero-cost edges between (copy) depots cannot be used, as targets are connected to (copy) depots by two edges in a feasible tour. Therefore an edge with larger cost than zero can be added which improves the lower bound. To conclude, this method gives a bound that is at most 50% from the optimal tour cost.

5.2.3 Lower bound on a complete tour (LC, MT)

The lower bound determined at v_p for small values of p is weak, because many edges in the MST are of cost zero. Therefore, the maximum of the bound on the full tour and the one determined at v_p is used. This section describes how to derive a bound on the full tour based on the fact that a target in a tour has degree two.

Assuming the tour is closed, i.e. there is an edge (of zero cost) between the depot and the last copy depot, each node in a tour has degree two: it has one ingoing and one outgoing edge. Let $v(u) = (v_1(u), ..., v_{n-1}(u))$ for $u \in V_T$ denote the ordered set of nodes that can be visited from u (so unequal to u itself), such that $v_1(u)$ denotes the node closest to u (recall that V_T is the set of nodes, i.e. targets and (copy) depots in the transformed graph G_T). A lower bound on the full tour is obtained by adding the two cheapest edges $v_1(u)$ and $v_2(u)$ for each node $u \in V_T$ and compensating for all edges that have been counted twice. Note that targets must be connected to depots in a feasible tour, so at least two edges incident to a depot or a copy depot are also incident to a target.

Recall the definition of the cost function c_T from Section 3.2. Let $T = V_T \backslash \{C_0, ..., C_p\}$ denote the set of targets in V_T. Let $w(u) = (w_1(u), ..., w_{|T|-1})$ for $u \in V_T$ denote the set of *targets* ordered by the edge cost of the edges incident to u, where $w_1(u)$ is the target closest to node u (so unequal to u itself). Let v_q denote the target that is closest to any of the depots, i.e. for which $\min_{i \in \{0,...,p\}} w(C_i)$ is smallest. A lower bound on the full tour is then given by:

$$\mathrm{LB}_{\mathrm{LC}}(u) = 2 \cdot \min_{i \in \{0,...,p\}} w(C_i) + \frac{1}{2} \sum_{u \in V_T \backslash \{v_q\}} c_T(v_1(u), u) + c_T(u, v_2(u)). \tag{5.1}$$

The factor $\frac{1}{2}$ compensates for all edges that have been counted twice. This bound can be improved by considering the outgoing edge of a (copy) depot.

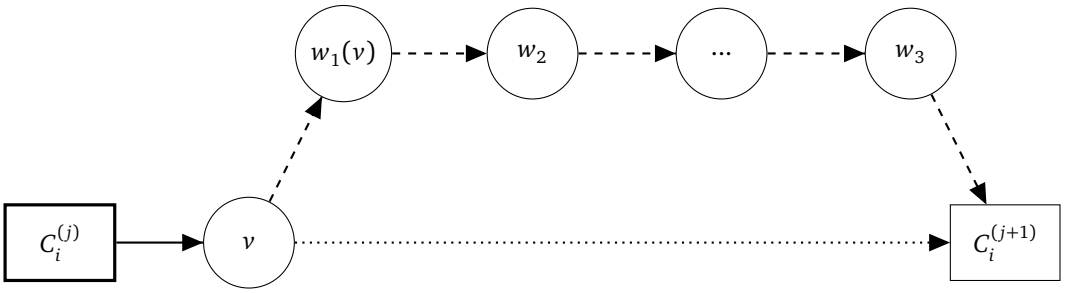


Figure 5.4 – Consider two possible scenarios for a salesman starting from (copy) depot $C_i^{(j)}$ and ending at $C_i^{(j+1)}$. A salesman either visits one target v and directly visits copy depot $C_i^{(j+1)}$ (dotted), or he visits at least two targets (at least $w_1(v)$ and possibly more) and then visits copy depot $C_i^{(j+1)}$ (dashed). Rectangular shapes represent (copy) depots, circular shapes represent targets.

Let $C_i^{(j)}$ denote a (copy) depot in the tour ($j = 0$ corresponds to the depot). Then, there are two options: starting from a (copy) depot, either exactly one target is visited or at least two targets are visited before the succeeding copy depot is passed, as shown in Figure 5.4.

First, determine a lower bound on the cost of visiting some targets in a connected tour. The bound is based on the idea that if a target is connected in a tour with multiple targets, then it is likely that it is connected to its closest target. To simplify notation, let w_2 denote the target closest to $w_1(v)$ that is not equal to v. Also, let w_3 denote the target closest to $C_i^{(j+1)}$ but not equal to w_1 and v, i.e. $w_2 = w_2(w_1(v))$ and $w_3 = w_2(C_i^{(j+1)})$. For the tour through multiple targets to be complete, target v must be connected to its closest depot, node w_3 can be connected to either a target or a (copy) depot and the tour must finish at a (copy) depot. Therefore, a lower bound on the cost of this tour is given by the sum of $c_T(v, w_1(v)) = \min_{w \in T} c_T(v, w)$ and $\min\{\min_{w \in T} c_T(w_3, w), \min_{w \in T} c_T(w, C_i^{(j+1)})\}$, plus a lower bound on the cost of moving from $w_1(v)$ to w_3 (possibly via multiple targets).

This lower bound is given by the cost of moving from $w_1(v)$ to w_2, plus the average of the cost of the shortest edge incident to w_2 and the shortest edge incident to w_3:

$$\mathrm{LB}_{w_1,w_3} = c_T(w_1(v), w_2) + \frac{1}{2} \min_{z \in T \setminus \{w_1\}} c_T(w_2, z) + \frac{1}{2} \min_{z \in T \setminus \{w_2\}} c_T(z, w_3).$$

A lower bound on the cost of visiting at least two targets in this tour starting from (copy) depot $C_i^{(j)}$, visiting target v and returning to either copy depot $C_i^{(j+1)}$ or another target is therefore given by:

$$\mathrm{LB}_{\mathrm{MT}}(v) = c_T(C_i^{(j)}, v) + f(v, w_3) + \min\{\min_{w \in T} c_T(w_3, w), \min_{w \in T} c_T(w, C_i^{(j+1)})\}, \qquad (5.2)$$

where $f(v, w_3) = c_T(v, w_1(v)) + \mathrm{LB}_{w_1,w_3}$ is a lower bound on the cost of moving from v, possibly through multiple targets, to w_3.

Then, this lower bound can be used if it is smaller than the bound LB_{LC}, (5.1), that connects all visited targets in this segment, where target v is individually assigned to a depot. Therefore, determine the cost of assigning node v to a depot twice and determine a lower bound on the cost of connecting targets $w_1(v)$, w_2 and w_3 to any of the nodes in the problem. This can be done by the MST bound described in the previous section or by the LC bound (5.1). If this bound is smaller than the bound that restricts node v to be connected to at least one more target, this smaller bound is used to connect the set of targets. The targets may still appear together in the tour but then LB_{LC} serves as a lower bound.

However, multiple segments where multiple targets are connected may appear in an optimal tour. Therefore, determine $LB_{MT}(v)$ and $LB_{LC}(v)$ for each target v, check whether $LB_{MT}(v) < LB_{LC}(v)$ and if so, keep track of the segments obtained this way and the corresponding costs $f(v, w_3)$. Remove all duplicate segments and if targets appear in multiple segments, keep the segment with largest cost so that each target appears at most once in a segment. For the lower bound, the obtained segments can either all be connected to copy depots, or multiple segments may form one larger segment and are then connected to copy depots. The option of least cost is chosen.

The segments are now all connected to depots, but not all targets have necessarily been visited. Therefore, determine a lower bound by (5.1) for the targets that are not in any of the segments and add it to the total cost of visiting all determined segments.

The bound obtained by the above procedure may be weaker than the MST or LC bound, which mainly depends on the quality of the bound (5.1) for the targets that most likely do not occur together in a tour. It also depends on the quality of the bound LB_{LC}: the bound (5.2) may be rejected if the bound LB_{LC} is weak, even though the targets may appear in a connected tour together.

Chapter 6

Branch and bound method for non-redundant tours

Regularly, cables break during road works. If the two cables connecting a street cabinet are in different trenches, a damage to one of the cables will not immediately affect the customer: the data transfer can be rerouted through the undamaged cable. Such a connection is called *redundant*. If both cables are in the same trench, which is called a *non-redundant* or *single-sided* connection, many customers remain disconnected until the cable is repaired. Recall Figure 3.4b for an example of a redundant and a non-redundant connection.

Costs can be reduced when non-redundant connections are allowed, but at the price of a less reliable network. To reduce the risk of network failure, KPN allows at most 2000 customers subject to a non-redundant trench. Comparing redundant tours to tours where single-sided connections are allowed gives insights in the extra costs of building a more reliable network.

In Section 6.1, an approach for solving the above described problem is presented. A branch and bound (B&B) method is used to find the optimal non-redundant tour. To make the algorithm efficient, duplicate and infeasible tours should not be traversed. How to eliminate these paths is shown in Section 6.2. Some properties of the cost function are described in Section 6.3 and are used in Section 6.4 to show that a non-redundant tour is always cheaper than a redundant tour. Finally, Section 6.5 and Section 6.6 describe upper and lower bounds that are used to determine whether the branches emerging from a path do definitely not contain the optimal solution, so that traversal of this path is stopped.

6.1 Solution approach

In a non-redundant connection, data must travel from a depot to some targets and then back to the depot through the same trench. The path from a target to a depot was defined to be the smallest one, which means that the path from the last target back to the depot is not necessarily the already used trench. By visiting the targets again but in reverse order, the cable is forced to lay in this trench. In a redundant connection, each target is visited exactly once. Therefore, tours obtained from the heuristic model of Chapter 4 and the exact branch and bound (B&B) method of Chapter 5 are redundant.

Note that a path connecting exactly one target to a depot is a special case, since the same trench is used twice even though all nodes are visited exactly once. Such a connection is allowed in redundant and in non-redundant tours.

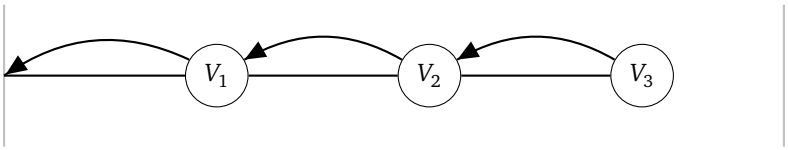


Figure 6.1 – A schematic view to scale of a problem containing three targets and two depots (grey lines). Start from the cable on the left and visit the targets V_1, V_2 and V_3. To return to the cable from V_3 using the currently dug open trench, visit the targets again in reverse order: the cable closest to V_3 is not using the trench that is dug open and will create a redundant tour.

Recall from Section 3.4 that $P_i^{(j)} = (V_{j_1}, V_{j_2}, ..., V_{j_{r_j}}, C_i^{(j)})$ denotes a segment in a redundant tour, containing $r_j \geq 1$ targets and exactly one copy depot of type i. The tour allowing single-sided connections corresponding to segment $P_i^{(j)}$ is given by $\tilde{P}_i^{(j)}$, which consists of a depot and $\tilde{m}_j$ copy depots of type i, r_j targets (the ones that are in $P_i^{(j)}$) and r_j copy targets. The cost function $\tilde{c}_T$ has been defined in Section 3.4. The copy targets allow a street cabinet to be visited twice.

Let $\tilde{p}_i^{(k)}$ denote a segment in $\tilde{P}_i^{(j)}$ that contains $\tilde{r}_k$ targets, for $1 \leq k \leq \tilde{m}_j$, so that the segment consists of some targets and its copies, where the first target is incident to copy depot $C_i^{(k-1)}$, and ends with copy depot $C_i^{(k)}$. Since the tour $\tilde{P}_i^{(j)}$ contains $\tilde{m}_j$ copy depots, it consists of $\tilde{m}_j$ segments and therefore $k \leq \tilde{m}_j$. See Figure 6.2 for the structure of a redundant and a non-redundant segment $\tilde{p}_i^{(k)}$ that contains $\tilde{r}_k$ targets in $\tilde{P}_i^{(j)}$.

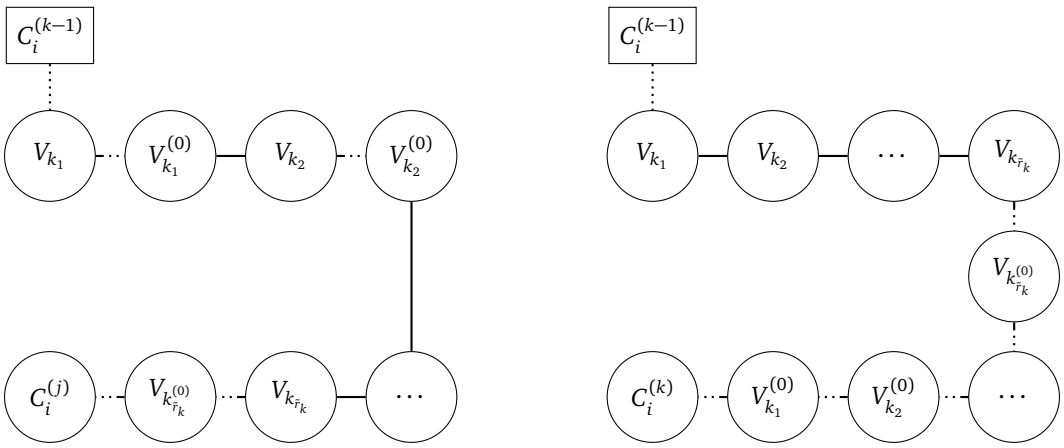


Figure 6.2 – A redundant (left) and a non-redundant (right) segment $\tilde{p}_i^{(k)}$ in $\tilde{P}_i^{(j)}$ that contains $\tilde{r}_k$ targets. All dashed-dotted edges represent zero-cost edges. Copy depot $C_i^{(k-1)}$ is not in this segment; however it is included in the costs of this segment.

Note that the segment $P_i^{(j)}$ and the redundant representation $\tilde{P}_i^{(j)}$, as shown in Figure 6.2, are of the same cost: the edge between a copy depot and the first target is present in both tours, the edges $(V_{k_l}, V_{k_{l+1}})$ and $(V_{k_l}^{(0)}, V_{k_{l+1}})$ are of equal cost and all

other edges are of cost zero.

Each segment $\tilde{P}_i^{(j)}$ is optimised independently of the other segments by a B&B method. As stated in Chapter 5, all possible permutations of the nodes in a segment emerge from the root of a tree and are then evaluated systematically. A natural choice for the root is the depot C_i for a segment $\tilde{P}_i^{(j)}$. Just as in the previous chapter, the tree is explored in a depth-first search order, as shown in Figure 5.1. The children of a node are explored in ascending order of edge cost.

6.2 Infeasible tour and duplicate elimination

Traversing duplicate paths and infeasible tours in the tree is time consuming and does not contribute to finding an optimal tour. Therefore, they should not be traversed. First, infeasible paths are discussed and then restrictions are given that make sure that duplicate paths are not traversed.

In a feasible tour, if a segment contains one or more targets, all corresponding copy targets are also in the segment. The reverse must also hold: if a copy target is in the segment, its corresponding target must be in there. Without loss of generality, assume that targets are visited before their copies. Moreover, assume that copy depot $C_i^{(j)}$ is visited before $C_i^{(j+1)}$. Therefore, the following restrictions aply:

- Visit a target before its corresponding copy.
- Do not visit a copy depot when a target is in the current segment, but the copy target is not.
- Visit copy depot $C_i^{(j)}$ before $C_i^{(j+1)}$, possibly visiting targets in between. Moreover, two copy depots are traversed directly succeeding each other only if all targets and copy targets have been visited.

Preventing the traversing of infeasible paths and some duplicates has been discussed above. The elimination of the remaining duplicate paths is now discussed. Consider a tour $\tilde{P}_i^{(j)}$ with $\tilde{p}_i^{(1)}$ as the first segment. The tree contains many branches that emerge from the path $(C_i, \tilde{p}_i^{(1)})$, containing all remaining targets and copy depots. Then, all tours that contain this segment $\tilde{p}_i^{(1)}$, but not as the first segment (where the copy depot has been renumbered), are a duplicate of a path that emerges from $(C_i, \tilde{p}_i^{(1)})$. Consider Figure 6.3 as an example: all paths that contain the segment $(V_1, V_2, V_2^{(0)}, V_1^{(0)}, C_i^{(j)})$ for any $j > 1$ are a duplicate of the outer left path. To conclude, each segment in a given tour can appear anywhere in the tour which leaves $\mathcal{O}(s!)$ duplicates in the tree, where s denotes the number of segments in the path.

Duplicates can be eliminated by forcing exactly one target that has not yet occurred in the current path to appear in the next segment. This target is chosen to be the first target that is explored from the parent node and therefore is the node that has the least-cost edge of all the children of this parent node. This target does not necessarily appear as the first target of the segment, as that would eliminate too many paths.

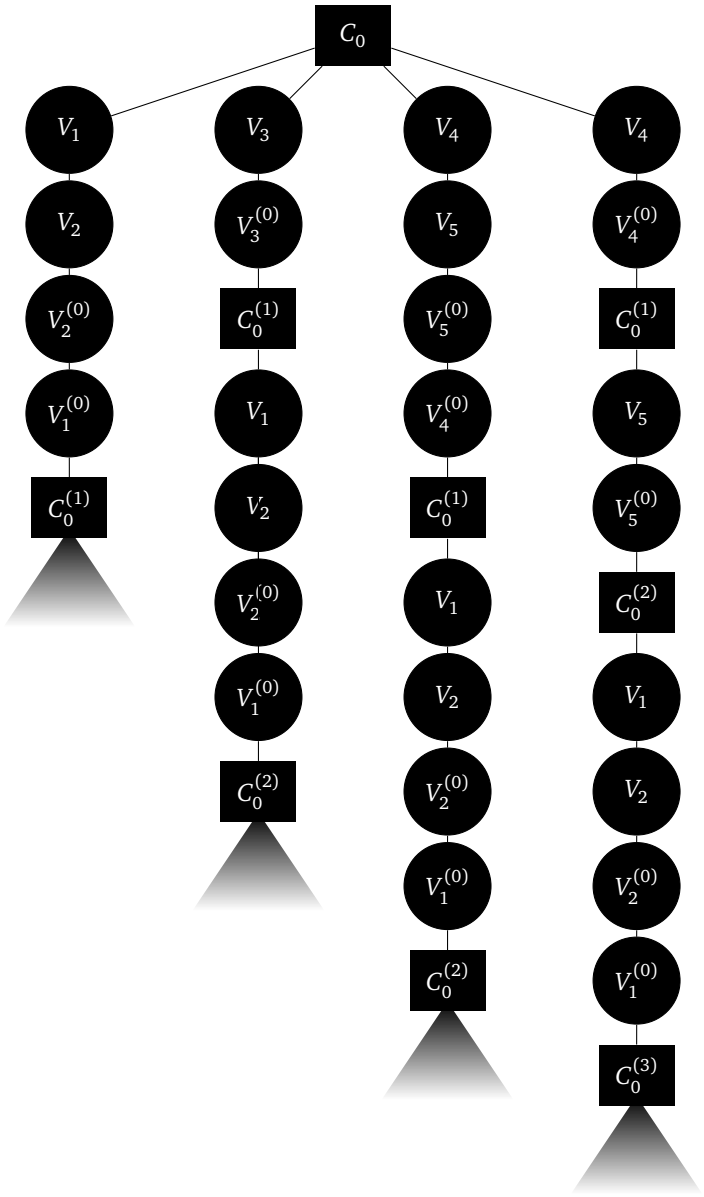


Figure 6.3 – The outer left path starts with the segment $(V_1, V_2, V_2^{(0)}, V_1^{(0)}, C_0^{(1)})$ and all possible paths that include this segment emerge from $C_0^{(1)}$. Some paths that include this segment, but not as the first, are also visualised. All these paths occur as a branch of the first path and should therefore be eliminated.

6.3 Cost function

Recall from Section 3.4 that the cost function depends on the tour constructed so far. Moreover, recall that the cost of a segment includes the cost of the edge connecting a (copy) depot to the first target, even though this (copy) depot is not included in $\tilde{p}_i^{(j)}$. This section describes some properties of the cost function during tree traversal.

There are three types of nodes that can be visited from the root of the tree: a node is either a target, a copy target or a copy depot. The claim is that the cost of edges between targets and between a target and a (copy) depot is larger than zero (unless the physical distance between a depot and a target is zero). All other edges that are traversed are of zero cost. Let us consider the three possiblities for the node that is visited.

Target: Whenever a target is visited, the trench is dug open for the first time and therefore the cost of an edge concerning a target is always larger than zero.

Copy target: The visited node is a copy target $V_k^{(0)}$ corresponding to target V_k. This target is already in the path, because targets are visited before their copies. Then, either the previous node is a copy target, or it is the target V_k.

In case the previous node is V_k, then the cost of the edge $(V_k, V_k^{(0)})$ is zero by definition. In case the previous node is a copy target, denote it by $V_l^{(0)}$. The node V_l is already in the path. Then, if the edge (V_l, V_k) is in the path, the edge $(V_l^{(0)}, V_k^{(0)})$ is of cost zero. In Section 6.4 it is shown that the optimal tour is a non-redundant tour and therefore the order of the copy targets is the reverse of the order of the targets. Therefore, copy targets are only visited when the edge concerning the corresponding targets is in the tour. Therefore, the traversed edges between copy targets are of cost zero.

Copy depot: The new node is a copy depot. There are two possibilities: the other endpoint of the new edge is another copy depot, or is a copy target. The other endpoint cannot be a target, since a copy target must be added before adding a copy depot and targets are always added before their corresponding copy targets. The cost of edges between succeeding copy depots is zero by definition. Moreover, the cost between a copy target and a copy depot is also zero, since the edge between the preceding copy depot and the target corresponding to the copy target must already be in the path.

To conclude, only edges concerning targets are not zero-cost edges: all visited edges where the last added node is either a copy target or a copy depot are of cost zero.

6.4 Redundant vs. non-redundant tours

In a redundant path, a copy target is visited directly after the corresponding target. In a non-redundant path, one or more other targets are visited before the corresponding copy target is visited for all targets but one, recall Figure 6.2.

In Section 6.4.1, a comparison is made between the cost of a redundant and a non-redundant tour. It is shown that a redundant segment $\tilde{p}_i^{(k)}$ in the tour $\tilde{P}_i^{(j)}$ is never optimal when non-redundant connections are allowed. Many paths can be eliminated high up in the tree, which results in a more efficient B&B method. Consequently, restrictions on visiting nodes are given in Section 6.4.2. The cost function $\tilde{c}_T$ has been defined in Section 3.4 and its properties have been discussed in Section 6.3.

6.4.1 Comparison of tour costs

Let $(V_{k_1}, V_{k_1}^{(0)}, V_{k_2}, V_{k_2}^{(0)}, ..., V_{k_{\tilde{r}_k}}, V_{k_{\tilde{r}_k}}^{(0)}, C_i^{(1)})$ denote a redundant segment with $\tilde{r}_k$ targets, the same number of copy targets and a copy depot of type i. Recall that the cost of a segment includes the cost of moving from the depot of type i to the first target. The cost of this segment is given by:

$$c_1 = \tilde{c}_T(C_i, V_{k_1}) + [\sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}, V_{k_l}^{(0)}) + \tilde{c}_T(V_{k_l}^{(0)}, V_{k_{l+1}})] + \tilde{c}_T(V_{k_{\tilde{r}_k}}^{(0)}, C_i^{(1)}), \tag{6.1}$$

$$= \tilde{c}_T(C_i, V_{k_1}) + \sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}^{(0)}, V_{k_{l+1}}) + \tilde{c}_T(V_{k_{\tilde{r}_k}}^{(0)}, C_i^{(1)}). \tag{6.2}$$

The edge between a target and its copy is zero, so the first term in the summation of (6.1) is removed to obtain (6.2).

Rearranging targets and copy targets in the redundant segment yields the corresponding non-redundant segment, which is given by $(V_{k_1}, V_{k_2}, , ..., V_{k_{\tilde{r}_k}}, V_{k_{\tilde{r}_k}}^{(0)}, ..., V_{k_2}^{(0)}$, $V_{k_1}^{(0)}, C_i^{(1)})$, which is of cost:

$$c_2 = \tilde{c}_T(C_i, V_{k_1}) + [\sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}, V_{k_{l+1}}) + \tilde{c}_T(V_{k_{l+1}}^{(0)}, V_{k_l}^{(0)})] + \tilde{c}_T(V_{k_{\tilde{r}_k}}, V_{k_{\tilde{r}_k}}^{(0)}) + \tilde{c}_T(V_{k_1}^{(0)}, C_i^{(1)}), \tag{6.3}$$

$$= \tilde{c}_T(C_i, V_{k_1}) + \sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}, V_{k_{l+1}}). \tag{6.4}$$

In (6.3), many edges are of cost zero: all edges between copy targets are zero, since the edges between the corresponding targets are also in the tour. Moreover, the edge $(V_{k_1}^{(0)}, C_i^{(1)})$ is of cost zero since V_{k_1} is connected to a (copy) depot of type i. This yields (6.4). Comparing costs (6.2) for a redundant tour and (6.4) for a non-redundant tour

yields:

$$\begin{aligned} c_1 &= \tilde{c}_T(C_i, V_{k_1}) + \sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}^{(0)}, V_{k_{l+1}}) + \tilde{c}_T(V_{k_{\tilde{r}_k}}^{(0)}, C_i^{(1)}), \\ &= \tilde{c}_T(C_i, V_{k_1}) + \sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}, V_{k_{l+1}}) + \tilde{c}_T(V_{k_{\tilde{r}_k}}^{(0)}, C_i^{(1)}), \\ &\geq \tilde{c}_T(C_i, V_{k_1}) + \sum_{l=1}^{\tilde{r}_k-1} \tilde{c}_T(V_{k_l}, V_{k_{l+1}}) = c_2. \end{aligned}$$

In the second line, realise that edges in a redundant tour between a copy target and another target are of the same cost as edges between two targets. Note that the inequality is strict when the physical distance between target V_{j_k} and a (copy) depot of type i is larger than zero. To conclude, by rearranging targets and copy targets in a segment of a redundant tour, a non-redundant tour of smaller, and at most of equal, cost can always be obtained. Restrictions on the nodes to visit during tree traversal so that redundant tours are not visited are given in Section 6.4.2.

6.4.2 Elimination of redundant tours

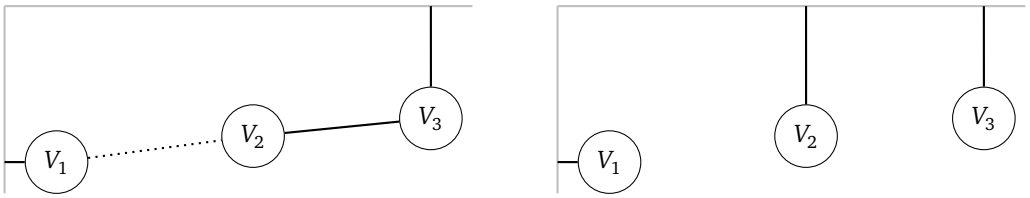
An optimal segment containing $\tilde{r}_k$ targets subject to depot i is as follows: $(V_{k_1}, V_{k_2}, .., V_{k_{\tilde{r}_k}}, V_{k_{\tilde{r}_k}}^{(0)}, ..., V_{k_2}^{(0)}, V_{k_1}^{(0)}, C_i^{(k)})$. That means that a target can only be followed by its copy or by another target. If a target is followed by its copy, then the order of the other copy targets in this segment is fixed and no other target can be visited in this segment. Therefore, in addition to the constraints on traversing the tree given in Section 6.2, the following restrictions apply:

- Do not visit a target after a copy target, so that only non-redundant segments are traversed.
- If the current node is a target, the only valid copy target to visit is the corresponding one.
- If the current node is a copy target, visit the copy target or copy depot that connects through a zero-cost edge.

Then, only non-redundant tours are traversed.

6.5 Upper bound on the tour cost

A strong upper bound is necessary for an efficient B&B method, as nodes need to be pruned as high up in the tree as possible. Two upper bounds are discussed: one that uses the redundant segment that is being optimised and one that uses a heuristic to obtain a feasible non-redundant tour. Both of them are discussed below. The upper bound that is used in the B&B algorithm is the minimum between the two bounds proposed above.



(a) Removing the largest edge from the redundant tour yields an upper bound.

(b) A stronger bound is obtained by individually assigning targets to depots.

Figure 6.4 – A schematic view to scale of an optimal redundant tour $T_T = (C_0, V_1, V_2, V_3, C_0^{(1)})$ that connects targets V_1, V_2 and V_3 to the cable of type i (grey lines). The edge of largest cost in the tour is (V_1, V_2). Also, $c_T(V_2, V_3) > c_T(C_0, V_2)$.

6.5.1 Upper bound from a redundant tour

As shown in Section 6.4, a non-redundant tour is at most of equal cost as the redundant tour when rearranging targets and copy targets. However, when copy depots are allowed to change places too, an even shorter non-redundant tour can be obtained.

A cheaper non-redundant tour can be obtained by removing exactly one edge in the redundant tour (and rearranging the nodes, including the copy depots, accordingly), regardless which edge. Consider for example the redundant tour that connects three targets in Figure 6.4a: removing any of these edges leaves a feasible, non-redundant tour. If both endpoints of the removed edge are targets, two segments are created that are both non-redundant. On the other hand, if one endpoint is a copy depot, the tour consists of one non-redundant segment. Therefore, to find an upper bound on the tour cost, subtract the cost of the edge of largest cost from the cost of the redundant tour.

6.5.2 Upper bound from heuristic method

The bound from the previous section is a weak bound if multiple targets that are redundantly connected are individually assigned to depots in an optimal non-redundant tour. Again, consider the example visualised in Figure 6.4. The largest edge is (V_1, V_2), so the upper bound found from removing the largest edge is given by $\tilde{c}_T(C_0, V_1) + \tilde{c}_T(V_2, V_3) + \tilde{c}_T(V_3, C_0)$. However, since $\tilde{c}_T(C_0, V_2) < \tilde{c}_T(V_2, V_3)$, a stronger bound for a non-redundant problem is obtained if the targets are assigned to a tour individually, which yields a cost $\tilde{c}_T(C_0, V_1) + \tilde{c}_T(C_0, V_2) + \tilde{c}_T(C_0, V_3)$ (all other edges are of cost zero) (see Figure 6.4b).

This section proposes an upper bound based on the Clarke and Wright savings heuristic [5], which is a method that creates multiple tours by assigning each target to its own depot and attempts to improve the tour by merging them.

Let $\tilde{m}_j$ denote the number of targets visited in a segment $\tilde{P}_i^{(j)}$. Then, assign each target and its corresponding copy target in this segment to its own copy depot. The resulting tour is $\tilde{T}_T = (C_0, V_{j_1}, V_{j_1}^{(0)}, C_0^{(1)}, ..., C_0^{(\tilde{m}_j - 1)}, V_{j_{\tilde{m}_j}}, V_{j_{\tilde{m}_j}}^{(0)}, C_0^{(\tilde{m}_j)})$. Let $\tilde{p}_i^{(k)}$ denote a segment in this tour. The cost of a segment $\tilde{p}_i^{(k)}$ in this tour is $\tilde{c}_T(C_0, V_{j_k})$.

The goal is to optimise the tour $\tilde{T}_T$ by merging segments. Recall that only non-redundant tours are considered. Therefore, a merged tour is constructed by adding the targets and copy targets of a segment in the middle of another segment, as illustrated in Figure 6.5.

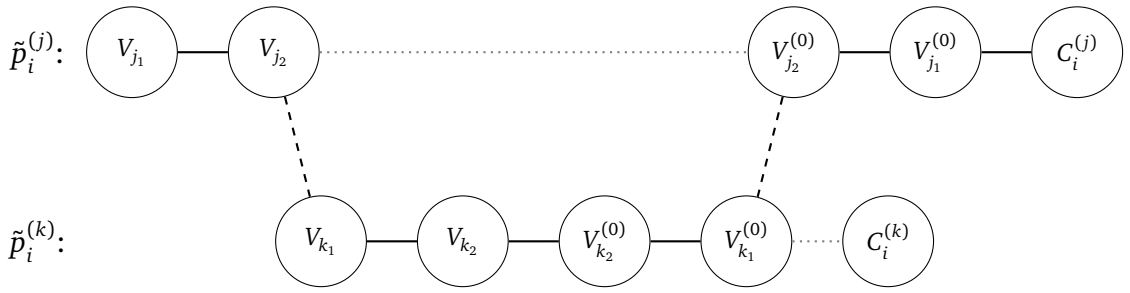


Figure 6.5 – Merging of a segment $\tilde{p}_i^{(j)}$ and $\tilde{p}_i^{(k)}$. The grey dotted edges have been removed from the original segments and the dashed edges have been added.

Merging two tours only results in an improved tour if the difference between the edge cost of the removed and added edges is positive. Therefore, the difference $\tilde{c}_T(C_i, V_{k_1}) - \tilde{c}_T(V_{j_2}, V_{k_1})$ must be positive for an improvement (note that the other removed and added edge are of cost zero). Tours are merged until no improvement is found.

6.6 Lower bound on the tour cost

This section describes how to obtain a lower bound on $\tilde{P}_i^{(j)}$, first up front and then during tree traversal in order to prune nodes in a path that does not lead to an optimal tour high up in the tree. During traversal, the maximum of the two bounds is used as lower bound.

6.6.1 Lower bound on the full tour

As described in Section 6.3, an edge between two targets or between a target and a (copy) depot has a cost larger than zero (unless the cost of the edge between a target and a copy depot is zero). By the definition of the TSP, each target in the segment must be visited and therefore each target has an incoming edge of a cost larger than zero. Also, at least one copy depot must be visited to make the tour feasible.

A lower bound is then given by the edge of least cost between a target and the depot, plus the edge of least cost between one of the remaining targets and the closest node, for each remaining target. The closest node can be a copy depot or a target. The edge between a target and its corresponding copy is zero, just as edges between copy targets, so we do not take copy targets into account.

Recall that V_T is the set that consists of the copy depot and the set of targets in $P_i^{(j)}$ (and thus also in $\tilde{P}_i^{(j)}$) and that r_j is the number of targets in $\tilde{P}_i^{(j)}$. Let j_q denote the

index of the target in $P_i^{(j)}$ that is closest to a depot. Then, a lower bound is given by:

$$\tilde{c}_T(C_i, V_{j_q}) + \sum_{k \in \{1,..,q-1,q+1,...,r_j\}} \min_{u \in V_T \setminus \{V_{j_k}\}} \tilde{c}_T(u, V_{j_k}).$$

6.6.2 Lower bound during tree traversal

A similar bound as the one derived above can be used during tree traversal. The first edge in the tree connects the depot to a target, so the feasibility criterion that a tour must contain at least one depot is satisfied.

Let us consider a path containing the nodes v_i, for $1 \leq i \leq p$, that is traversed up to level p. The index of the nodes corresponds to the level in the tree. Let $T^{(p)}$ denote the set of unvisited targets at level p, the node v_p and the depot C_i.

Just as in Section 6.6.1, each unvisited target will be visited through a non-zero cost edge. These targets can be visited from any of the nodes in $T^{(p)}$, so a lower bound is given by the sum of the cost of exactly one outgoing least-cost edge for each target and accounting for the edges counted twice.

However, depending on v_p, this bound may be improved. Node v_p is either a copy depot, a target or a copy target. In the latter two cases, the cheapest valid edge is a zero-cost edge: they can both be connected to a copy target and do not increase the lower bound. However, if v_p is a copy depot, it must be connected to a target that has not yet been visited. Let j_q denote the index of a target in $T^{(p)}$ that is closest to the copy depot v_p. To conclude, a lower bound at level p is given by:

$$\begin{cases} \displaystyle\sum_{l \in \{1,...,|T^{(p)}|-2\}} \min_{u \in T^{(p)}} \tilde{c}_T(u, V_{k_l}) & \text{if } v_p \text{ is a (copy) target,} \\ \displaystyle\tilde{c}_T(v_p, V_{j_q}) + \sum_{l \in \{1,...,q-1,q+1,...,|T^{(p)}|-2\}} \min_{u \in T^{(p)} \setminus \{v\}} \tilde{c}_T(u, V_{k_l}) & \text{if } v_p \text{ is a copy depot.} \end{cases}$$

Chapter 7

Results

One of the goals of this thesis is to obtain a redundant, optimal tour through one or more targets and (copy) depots. To do so, the heuristic method described in Chapter 4 has been developed. Moreover, non-redundant connections were investigated, since they increase the vulnerability of the network but reduce costs. They are obtained by the model of Chapter 6, so that the costs of a redundant and a non-redundant tour can be compared.

In total, 2200 street cabinets need to be activated. Their locations are given, including the MA area they are in. Also, the locations of the three types of cables are given for each area. In order to determine the cost between nodes and to visualise the tours, the Python package OSMnx is used. OSMnx combines the NetworkX package, that includes several algorithms to analyse the structure of a network and some drawing tools, and Open Street Maps (OSM) data. The NetworkX package includes the Dijkstra algorithm [4], which is used to find the shortest path (for visualisation) and its distance (for the cost matrix) between targets and between targets and depots.

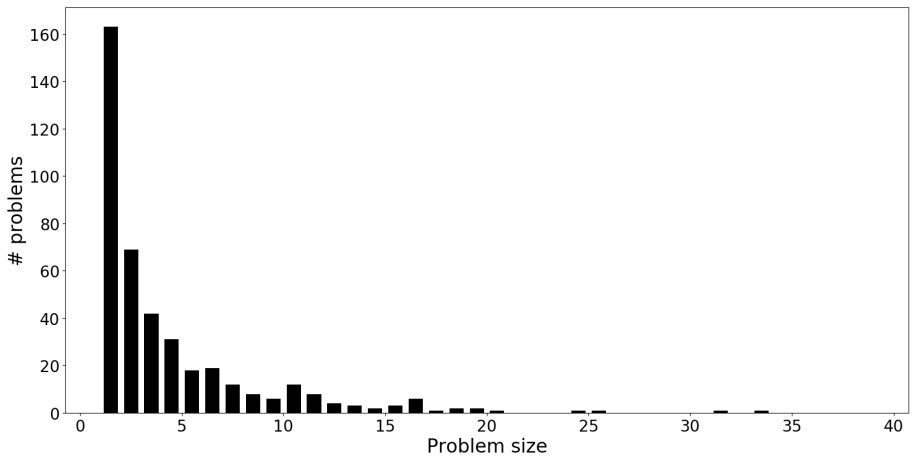


Figure 7.1 – Distribution of the number of street cabinets for each area.

For some targets, the distance to the nearest depot is zero, which means that the target is individually assigned to this depot in an optimal tour. Since these targets can be neglected, they have been removed from the problem sets. Therefore, a problem size denotes the number of targets where these targets have been removed. The distribution of the problem sizes is shown in Figure 7.1.

The quality of the heuristic method is examined in Section 7.1 by comparing the heuristic solutions to the optimal solutions found by the method described in Chapter 5. In order to incorporate non-redundant connections in the solutions obtained by the heuristic model, the branch and bound (B&B) method described in Chapter 6 is used. The quality of this method is investigated in Section 7.2.

7.1 Results for redundant tours

For each problem, twenty copy depots of type 0 and type 1 were added to the set of targets. The number of copy depots equals the number of segments in a tour. For all problems that contain at most twenty targets, all targets can now be individually assigned to a depot. Also, problems that have more than twenty targets have multiple tours in the optimal solution that connect more than one target (which can for example be checked by performing the MT bound in Section 5.2.3), so twenty copy depots is more than enough.

The heuristic algorithm starts from a heuristic solution, obtained by either a nearest neighbour, a farthest insertion or an arbitrary insertion method, recall Section 4.1. The algorithm then performs multiple iterations, where one iteration consists of an attempt to improve the tour by a 2- or a 3-OPT move starting from a randomly picked node in the tour, until no improvement has been found for any of the nodes as a starting node. Then a kick is performed to prevent being trapped in a local optimum. This process has been repeated ten times.

Although there is a difference in quality of the tours obtained by the tour construction methods for the same set of targets, the cost of the optimised tours is always the same. The quality of the heuristic method is determined by comparing the cost of the tours obtained by this model to the cost of the tours obtained by the B&B model described in Chapter 5. The tour that is optimised by the B&B method contains all targets, depots of types 0 and 1 and $x_i + 2$ copy targets of both types, where x_i denotes the number of segments subject to type i in the tour obtained by the heuristic method.

All problems that have up to eight targets are solved by the exact method; the other problems were too time consuming. For these problems, the cost of the tour obtained by the exact method equals the cost of the tour obtained by the heuristic model. For the remaining problems, the lower bounds discussed in Section 5.2 serve as a measure for the quality of the heuristic method. The quality of the lower bounds is discussed in Section 7.1.1. Moreover, the computation time of the heuristic and the exact method is discussed and compared in Section 7.1.2.

7.1.1 Lower bounds

Three types of lower bounds were used to determine the quality of the heuristic method: a bound based on a Minimum Spanning Tree (MST, Section 5.2.2), a bound based on the fact that all targets in a tour have degree two (LC, (5.1) in Section 5.2.3) and a method that is designed to improve bounds where multiple targets are connected in one tour (MT, Section 5.2.3).

Figure 7.2 shows the relative gap between the three different lower bounds that were used and the solution of the heuristic model. The gap between the MST bound and the tour cost obtained by the heuristic model varies between 10% and 50%. For some problems, the LC bound is strong, however for other problems, the gap is close to 50%. Moreover, the MT bound is the most accurate method for most problems, if a bound is obtained by this method. Out of the 150 problems, a MT bound is obtained for 47 problems. For 45 problems, this bound is stronger than the LC bound and for 43 problems, the bound is stronger than the MST bound. For all methods, the quality decreases as the problem size increases.

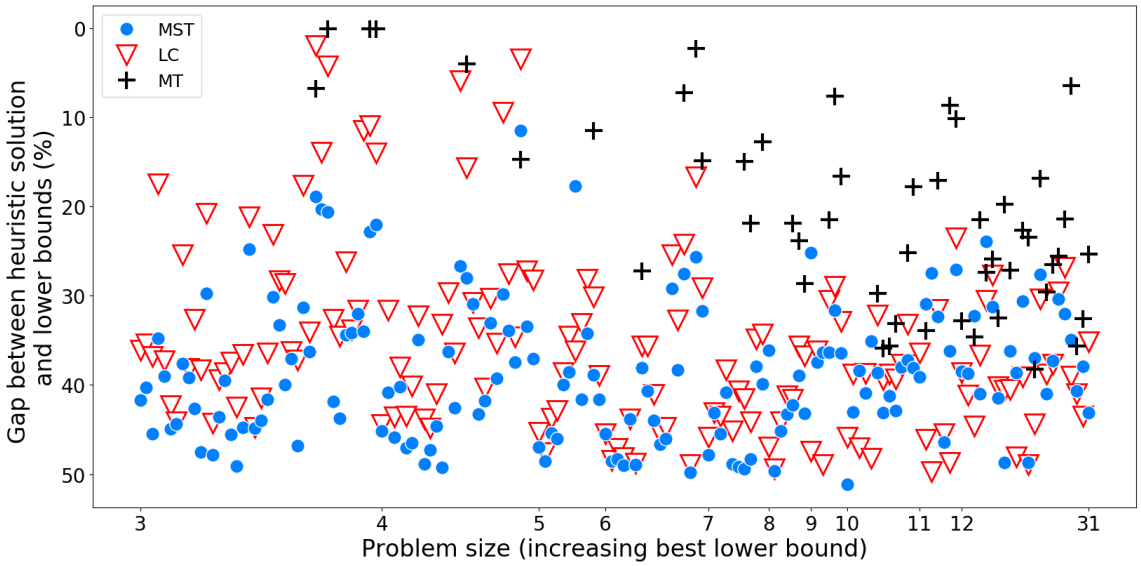


Figure 7.2 – Quality of the three types of lower bounds relative to the tour cost obtained by the heuristic model.

In general, all three methods have trouble finding an accurate lower bound for problems where targets have been assigned to copy depots individually in the optimal tour. The bounds for the cost of tours where the optimal segments contain multiple targets are more accurate. Below, the quality of the three methods is discussed.

Quality of MST bound

Consider Figure 7.3 for an example problem that contains five targets, two depots and four copy depots of each type. Assume that an individual assignment of the targets yields an optimal tour: the cost of this tour is $2 \cdot c_T(V_1, C_1^{(4)}) + 2 \cdot c_T(V_2, C_0^{(1)}) + 2 \cdot c_T(V_3, C_0^{(2)}) + 2 \cdot c_T(V_4, C_1^{(1)}) + 2 \cdot c_T(V_5, C_0^{(4)})$, which is the sum over ten non-zero cost edges. Since the problem contains fifteen nodes, an MST for this problem connects fourteen edges of least cost, which means that all (copy) depots are connected to each other (all zero-cost edges) and the targets are each connected to a copy depot by one edge (five edges). Now, the cost of the MST is exactly half of the cost of the optimal tour, as the targets should be connected to the (copy) depots by two edges in the optimal tour.

However, a feasible tour does not contain all edges between the depot and the copy depots. Therefore, one of these edges is removed so an edge of cost larger than zero

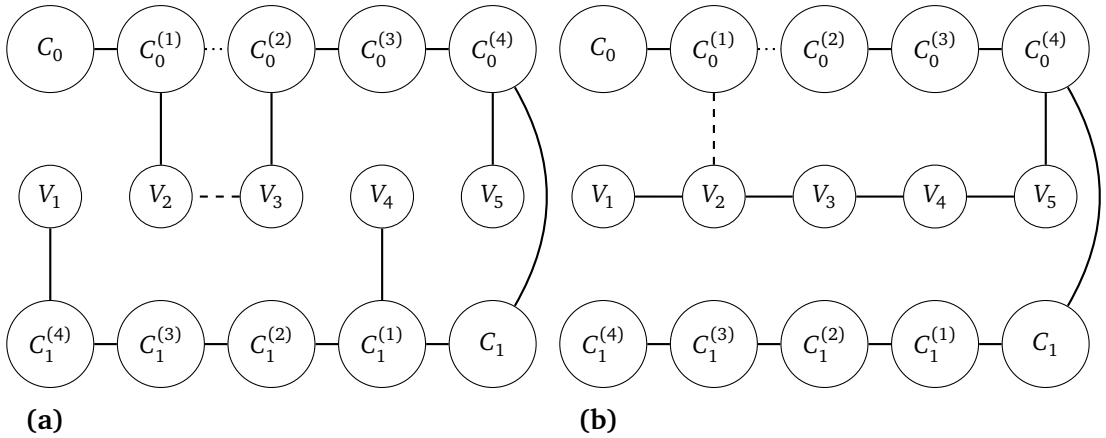


Figure 7.3 – A lower bound based on an MST for two types of problems: problems where targets are individually assigned to depots (left) and problems where multiple targets are directly connected to each other in an optimal tour (right).

that does not create a cycle can be added, recall Section 5.2.2. The one of least cost is chosen, which is for example (V_2, V_3). The most accurate bound for these type of problems is obtained if the distances between targets and between targets and (copy) depots is uniform. Then, the last added edge, in this example (V_2, V_3), increases the bound the most. Note however that only half (plus one) of the number of non-zero cost edges that are in the optimal tour are incorporated in this bound.

A more accurate bound is obtained if multiple targets are connected to each other in one tour. See Figure 7.3b for an example where the least-cost edge of a target is incident to another target. Assume that the optimal tour is given by $(C_0, V_1, V_2, V_3, V_4, V_5, C_0^{(1)}, C_0^{(2)}, C_0^{(3)}, C_0^{(4)}, C_0^{(5)}, C_1, C_1^{(1)}, C_1^{(2)}, C_1^{(3)}, C_1^{(4)}, C_1^{(5)})$, so the cost of this tour is given by $c_T(C_0, V_1) + \sum_{i=1}^{4} c_T(V_i, V_{i+1}) + c_T(V_5, C_0)$, which is the sum over six edges. Then, the MST contains all edges that connect (copy) depots with each other, and five of the six least cost edges between targets or between targets and copy depots. By adding the edge that makes the tour feasible, $(C_0^{(1)}, V_2)$ in this example, a stronger bound is obtained. Note that the optimal tour includes six non-zero cost edges, just as the MST bound.

To conclude, problems that connect single targets to a depot yield weak lower bounds, as the number of edges of cost larger than zero is only half (plus one) of the number of non-zero cost edges in the optimal tour. However, for problems that connect multiple targets in a tour, the number of non-zero cost edges in the MST bound equals the number in the optimal tour. Many problems are a combination of these two types of problems, which explains the variation in the bounds in Figure 7.2.

Quality of LC

This method uses the fact that all targets in a feasible tour have one incoming and one outgoing edge (apart from the first depot and the last copy depot). Moreover, at least two targets must be connected to a depot, as described in Section 5.2.3. The cost of the two least-cost edges are added for each of the remaining targets, accounting for duplicates.

For problems where all targets are individually assigned to a depot, the bound is accurate for (very) small problems but quickly becomes weaker as the problem sizes grow. See Figure 7.4a for the same example as in Figure 7.3a for an example of a tour where the edges of least cost from a target are incident to (copy) depots. Similar to the MST bound, only half of the number of non-zero cost edges (plus two) that are in the optimal tour are in the lower bound. Then, for small problems, the cost of the edge between a copy depot and a target dominates (which is (C_i, V_1) in this example), but as the problem grows, the bound becomes weaker.

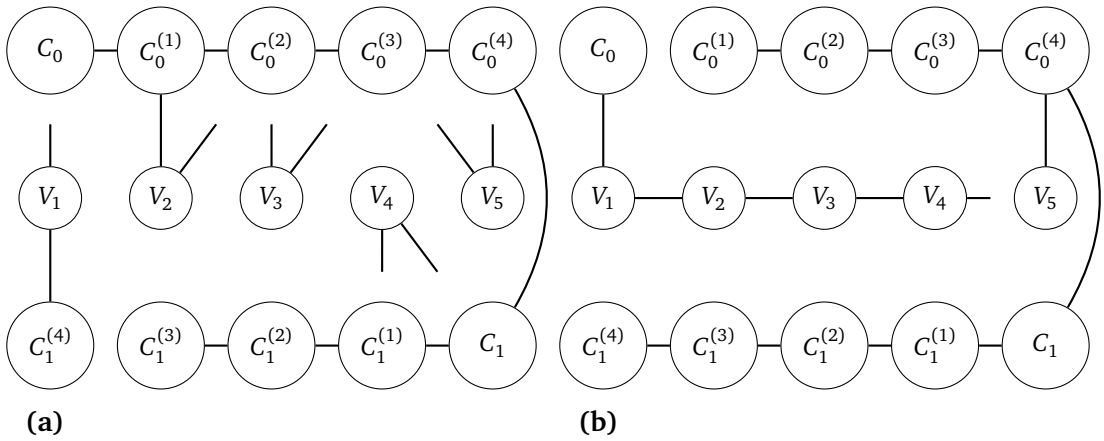


Figure 7.4 – A lower bound based on the fact that targets in a tour have degree two for two types of problems: a problem where all targets are individually assigned to a depot (left) and a problem where all targets are connected to each other (right) in an optimal tour. Note that a feasible tour always contains two edges from a (copy) depot to a target.

If the optimal tour consists of segments that connect multiple targets to a (copy) depot, the bound is more accurate. See Figure 7.4b (the same example as in Figure 7.3b) for a graph where the edges of least cost of a target are incident to another target. The quality of the bound decreases if the edges of least cost are not incident to other targets, but to depots. Therefore, for problems where the distance between targets is smaller than the distance between targets and the depots, the bound is strong.

Quality of MT

This bound was designed to obtain an accurate bound for problems where multiple targets are connected in one tour. For problems where the bound of connecting multiple targets in one tour to a depot is larger than the cost of assigning the first target to a depot individually plus the LC bound on connecting the remaining targets, this method does not yield a bound.

For most problems, the MT bound is an improvement compared to the MST and LC bound, as mentioned before. This bound uses the MST or the LC bound to determine the cost of targets that possibly do not occur together in a tour. Therefore, the quality of the bound is limited for problems that contain targets that have been assigned to a depot individually in the optimal tour, similar to the MST and LC bound.

7.1.2 Computation time

The tour that is being optimised is obtained by either a nearest neighbour, a farthest insertion or an arbitrary insertion method. As mentioned before, the cost of the optimised tours is the same. Also, there is no difference in computation time.

The average time necessary to perform the heuristic model for all problems is shown in Figure 7.5, together with the minimum and maximum times. The average computation time is calculated for each of the three types. The required time steadily increases as the problem sizes grow.

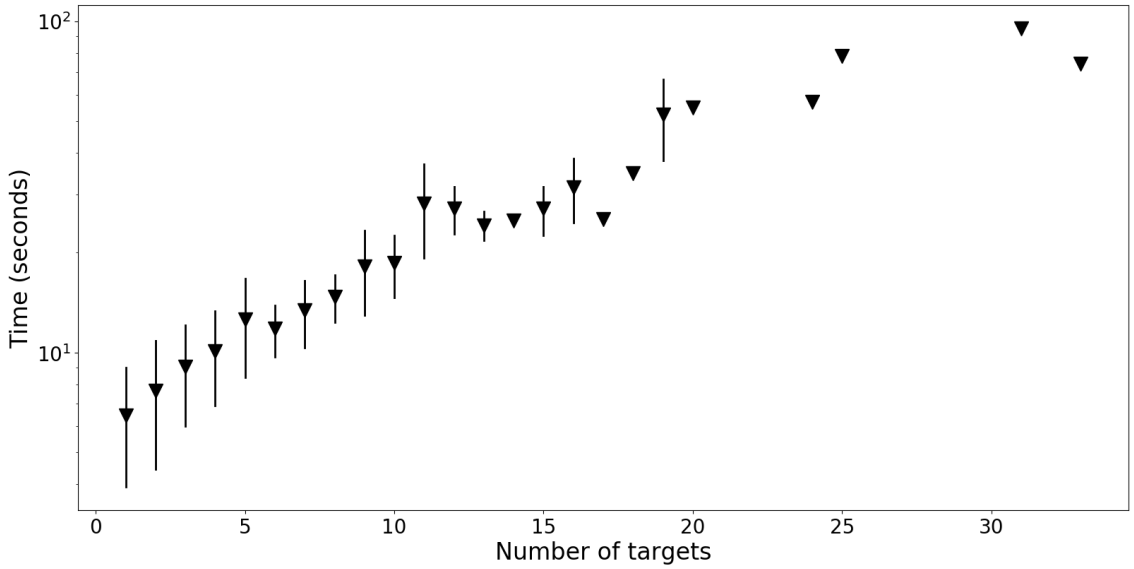


Figure 7.5 – The minimum, maximum and average values of the computation time required to finish the heuristic method for each problem size. The necessary time increases as the problem sizes grow.

The cost of the tours found by the heuristic method have been checked by the exact branch and bound method for redundant tours as explained in Chapter 5. As mentioned before, the exact method was too time-consuming for problems with more than eight targets. Figure 7.6 shows the time required for a problem to be solved by this method.

The tree has been traversed in a way such that the children of a parent node are explored in ascending order of the edge costs. In case the upper bound that was set up front is weak, a stronger one is hoped to be found quickly so that many nodes can be pruned early during tree traversal.

However, depending on the problem, the optimal tour may be found quicker if the children of a node are explored in descending order of edge costs. Then, when a branch is pruned (which is probably high up in the tree since the edge costs are high), all duplicates of the branches that emerge from the pruned node can be pruned right away and do not need to be traversed. In case nodes are visited in ascending order of edge costs, these paths have to be traversed deeper into the tree before they are pruned. This is an elimination of duplicates in the tree as shown in Figure 5.2b.

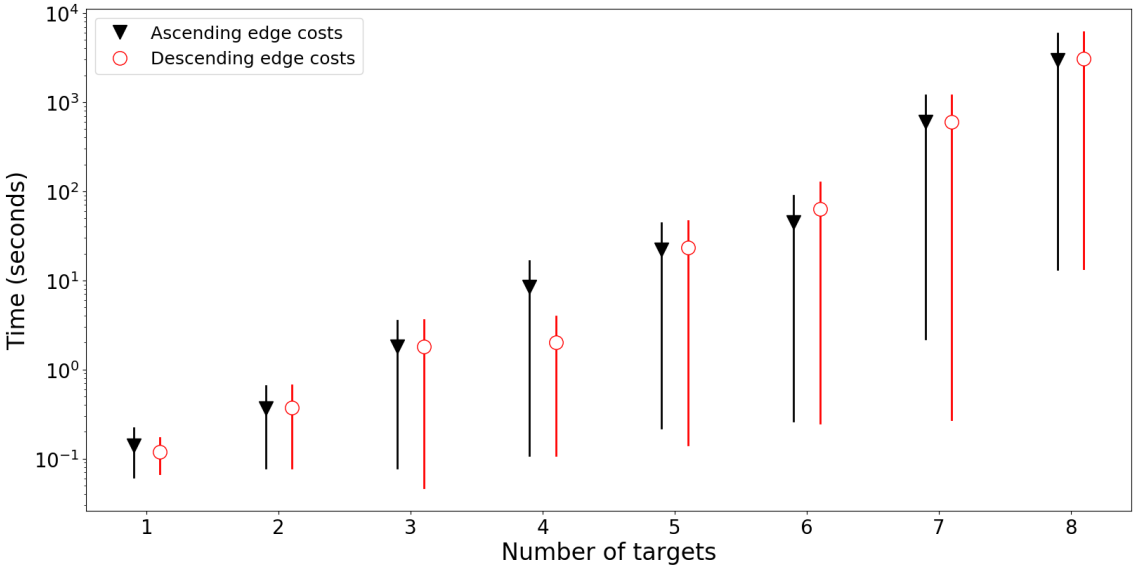


Figure 7.6 – Minimum, maximum and average values for the computation time of the exact method. As the problem size increases, the required time rapidly increases. The method has been investigated for trees where child nodes are traversed in ascending and in descending order of edge costs.

As is shown in Figure 7.6, the minimum and maximum values vary as the problem sizes increase. The average computation times are similar for both approaches. It depends on the type of problem which method is faster. If there are multiple clusters of targets in a problem far from a depot, and thus the distance between targets is not uniform, then the traversal in descending order is faster because of the duplicate elimination explained above. However, if the distance between the targets is uniform, a stronger upper bound is beneficial.

To conclude, the heuristic model yields the exact solution for problems with up to eight targets. The gap between the lower bounds and the exact solutions are similar to the bounds for larger problems. Therefore, the heuristic model yields promising results, especially considering the shortcomings of the lower bounds. Moreover, the time it takes for the heuristic model is acceptable, also for the large problems, while the exact method runs out of time.

7.2 Results for non-redundant tours

All segments in the redundant tours found by the heuristic method are optimised independently by the exact B&B method described in Chapter 6 to obtain optimal non-redundant tours. This yields 1506 problems. The quality of the upper and lower bounds derived in Section 6.5 and Section 6.6 respectively are considered by comparing them to the optimal tour in Section 7.2.2. The upper bound is the minimum of a bound obtained by removing the largest edge from a redundant tour (Section 6.5.1) and a bound obtained by the Clarke & Wright savings heuristic (Section 6.5.2). The lower bound is the sum over the cost of one least-cost edge for each target, accounting for duplicates, where at least one edge is connected to a depot.

7.2.1 Quality of the upper bounds

The performance of the two upper bounds is shown in Figure 7.7. Blue markers indicate problems where the derived bounds are the same, which was the case for 94, 36% of the problems. In case one of the two bounds performs better, the difference between the two bounds relative to the best upper bound, is shown.

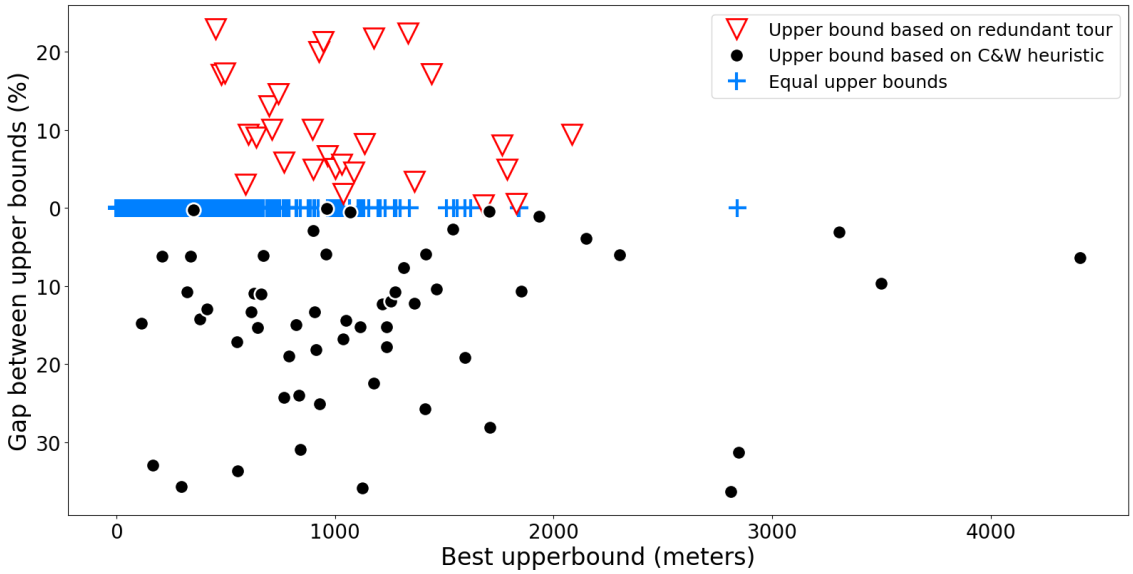


Figure 7.7 – Comparison of the upper bounds used relative to the smallest one.

The bound described in Section 6.5.1 performed worse than the bound described in Section 6.5.2 for 29 problems; the opposite holds for 56 problems. By using the method of Section 6.5.1, an improvement of at most 36, 35% has been obtained. Similarly, an improvement of 22, 99% has been obtained by using the method of Section 6.5.2.

Which bound yields the strongest result depends on the type of problem. For a problem where the targets are far from the depot, it is unlikely that a redundant tour will split into more than two non-redundant tours and therefore removing the edge of largest cost will result in a strong upper bound. For problems where targets are closer to a depot than to each other, redundant tours are likely to split up into multiple tours and therefore the method based on the Clarke and Wright savings heuristic will give a better result.

However, for many problems the bounds are equal: assigning targets to depots and merging the segments, or removing the edge of largest cost comes down to the same.

7.2.2 Quality of upper and lower bounds

This section investigates the quality of the upper and lower bounds for non-redundant tours. For 1439 problems, which is 95, 55% of the total number of problems, the derived upper and lower bound are equal. Therefore, the cost of the optimal tour has been found. For the remaining 67 problems, the upper and lower bounds are compared to the optimal tour found by the method of Chapter 6 in Figure 7.8.

First, consider the upper bound. In 40 of these 67 problems, the upper bound equals the cost of the optimal solution. The upper bound of the remaining 27 problems lies 4.67% from the optimal tour on average, with a maximum of 16.84%. Moreover, 90% of the upper bounds lie within 6% of the optimal tour cost.

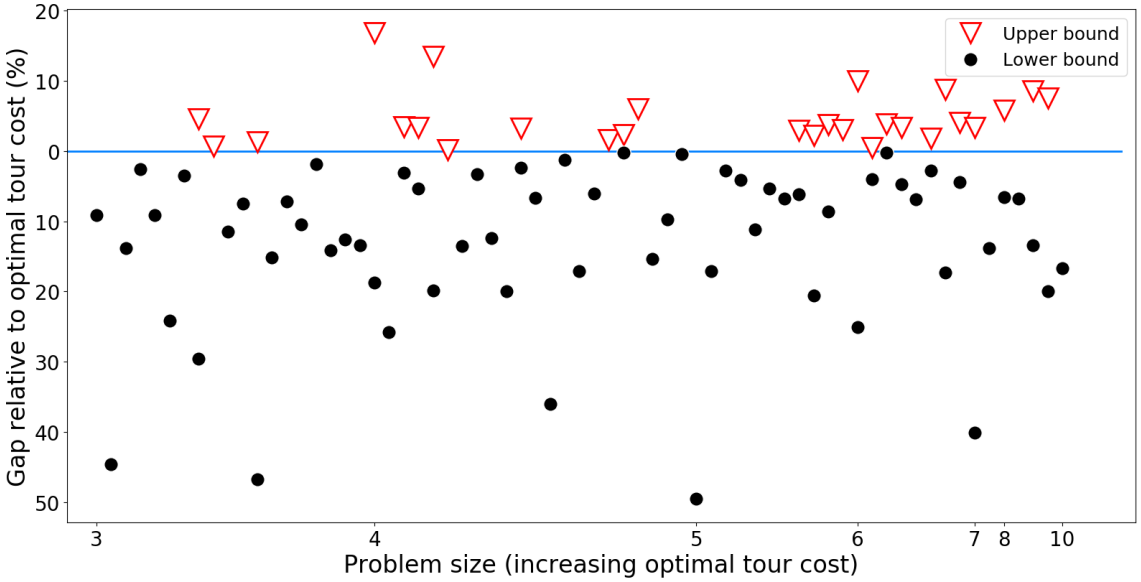


Figure 7.8 – The upper and lower bound for all problems where they are not equal to the optimal tour cost.

The maximum gap values are caused by problems where two or three edges can be removed to obtain a feasible non-redundant tour, when the targets are reordered. Figure 7.9 shows an example of a redundant tour $(C_i, V_1, V_2, V_3, V_4, C_i^{(1)})$. Removing the edges (V_1, V_2) and (V_3, V_4) and adding the edge (V_1, V_4) yields the optimal non-redundant tour. The obtained upper bound for this type of problem is not equal to the cost of the non-redundant tour, as the savings heuristic is unsuccesful and removing the largest edge does not yield this tour.

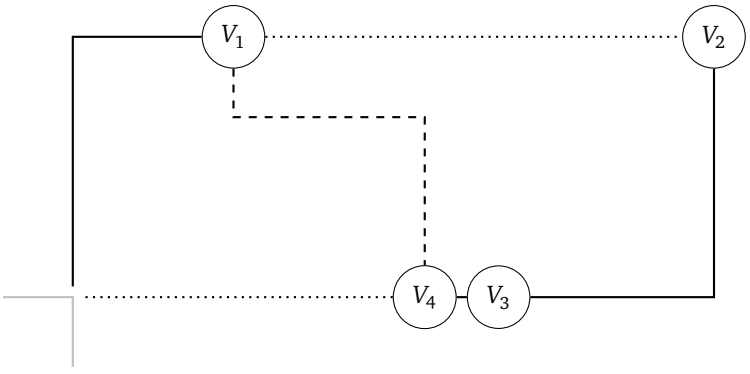


Figure 7.9 – An example to scale where four targets (V_1 to V_4) are redundantly connected to a cable (gray line). To obtain the optimal non-redundant connection, the edges (V_1, V_2) and (V_3, V_4) are removed (dotted) and the edge (V_1, V_4) is added (dashed).

Similar results can be obtained for the lower bound. For 63 problems, the lower bound does not equal the cost of the optimal solution. For these problems, the lower bounds

lie 12.08% from the optimal tour cost on average, where the maximum gap is 49.52%. Moreover, 85% of the lower bounds lie within 20% of the optimal tour cost.

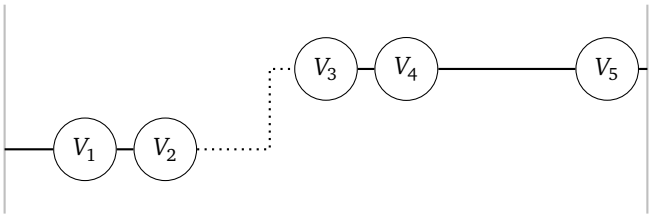


Figure 7.10 – An example to scale where five targets are redundantly connected to a cable (gray line). The optimal non-redundant tour is obtained by removing the edge (V_2, V_3). The lower bound for this problem is the sum of the cost of the edges (V_1, V_2), (V_3, V_4) and (V_5, C_i). The edge (V_4, V_5) is not incorporated in the bound, but does account for a large part of the tour cost.

The gaps between 40% and 50% are caused by problems that consist of two or three clusters of targets, where the distance between targets in a cluster is small, but the distance between clusters is relatively large. Also, the depots are relatively far for most clusters. See for example Figure 7.10: the bound uses the cost of the edges (V_1, V_2), (V_3, V_4) and (V_5, C_i). The edges that connect clusters in the optimal tour, as the depots are relatively far, are not incorporated in the bound but account for a large part of the tour cost.

7.2.3 Computation time

Figure 7.11 shows that the time necessary to complete a problem grows rapidly as the number of street cabinets grow. For two segments, containing 11 and 16 street cabinets, the method could not finish within 16 hours and was therefore forced to quit. All other problems have been solved.

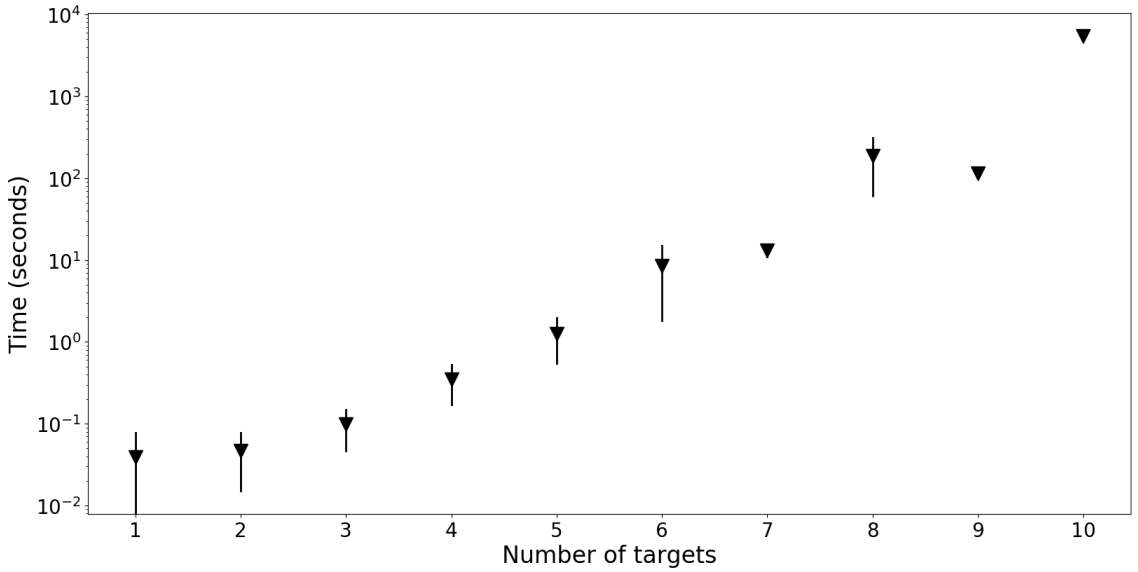


Figure 7.11 – Minimum, maximum and average values for the computation time for the B&B method for each problem size. As the problem size increases, the required time also increases. For a problem of eleven and sixteen targets, the problem could not be solved in acceptable time.

7.3 Redundant and non-redundant tours

As shown in Chapter 6, a non-redundant tour is always of lower cost than a redundant tour. The height of the extra costs of constructing a redundant tour instead of a non-redundant one are an argument for investing in a more reliable network. Figure 7.12 shows the investment that is necessary to obtain a redundant tour for each problem.

As the cost of the non-redundant tour grows, the necessary investment for obtaining a redundant tour also grows. For tours of large cost, the distance between targets and depots is large. At least one of the largest edges was removed from the redundant tour to obtain the non-redundant tour. Therefore, the difference between the cost of a non-redundant and a redundant tour grows as the cost of the non-redundant tour grows.

For many problems, the tour contains targets that are individually assigned to depots. For these targets, the cost of the redundant tour is twice the cost of the non-redundant tour. On the other hand, if the non-redundant tour contains multiple targets that are connected to the depot, the redundant tour is obtained by adding the edge that connects the proper target and the depot. Relative to the cost of the non-redundant tour, this cost is usually small.

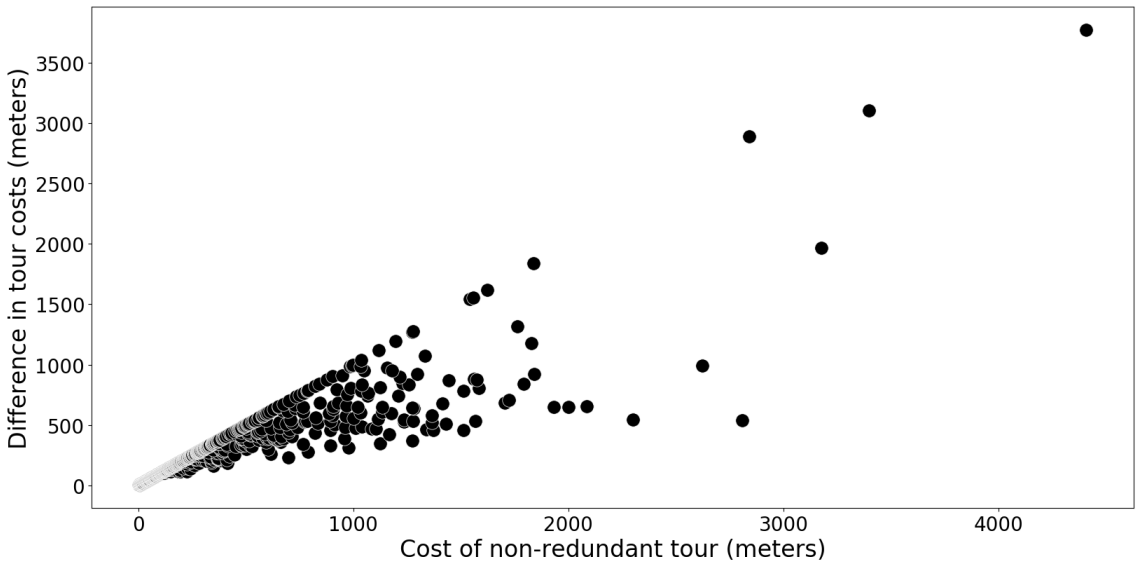


Figure 7.12 – Cost for the redundant and non-redundant tours.

Chapter 8

Conclusions and discussion

The goal of the performed research was to find an accurate cost approximation for connecting street cabinets to different types of cables in an optimal way. Another goal was to reduce work for the engineers, by supplying them with an optimal tour.

In order to do so, the heuristic model discussed in Chapter 4 was derived to obtain optimal redundant tours through a set of targets and (copy) depots. The quality of these tours was compared to that of the tours found by using the exact branch and bound (B&B) model described in Chapter 5, in case the problem sizes allowed for acceptable computation times. Moreover, non-redundant tours were obtained by the B&B algorithm described in Chapter 6.

The conclusions of this research are discussed below and are followed by directions for further research.

8.1 Conclusions

Tours obtained by the heuristic model for problems that have at most eight targets were compared to the solutions of the exact B&B algorithm for redundant tours and were found to be optimal. For larger problems, the computational requirements were unacceptable. The heuristic method solved all of the problems within acceptable time.

The quality of the tours obtained using the heuristic method is examined by considering three types of lower bounds: a bound based on a minimum spanning tree (MST), a bound based on the fact that all targets in a tour have degree two (LC) and a bound that is designed for accurate cost approximations of tours where multiple targets are connected in a single tour (MT). The quality was examined for all bounds, which resulted in the following findings:

- The quality is limited for problems in which targets are individually assigned to depots. In the worst case, the bound is half of the optimal tour cost.
- More accurate bounds are obtained for problems that connect multiple targets in a single tour.

Many problems contain both types of tours and therefore the quality of the lower bounds is mainly limited by the number of targets that are individually assigned to

depots. A gap larger than 50% by the MST did not occur for any of the problems, which is positive since a larger gap indicates that a problem was not solved optimally.

Non-redundant connections were also investigated. For the two largest problems, containing eleven and sixteen targets, the B&B method for non-redundant tours did not finish within eight hours and was therefore aborted. For all other problems, the method found the optimal non-redundant tour in acceptable time.

For these problems, the upper and lower bound give an accurate estimate of the cost of the non-redundant tour: for 96% the bounds were equal, meaning that the cost of the optimal tour has already been found. For the remaining problems, 90% of the upper bounds lie within 6% of the optimal tour cost and 85% of the lower bounds lie within 20% of the optimal tour cost. To conclude, for most problems of which the upper and lower bound are not equal, the upper bound is a more accurate estimate of the tour cost than the lower bound.

Lastly, the difference in costs between redundant and non-redundant connections were investigated, as costs are an important argument for improving the reliability of the network. Problems for which the cost of the optimal tour are dominated by targets that have been assigned to depots individually require a relatively large investment. On the other hand, problems where multiple targets are connected in one tour require a relatively smaller investment.

8.2 Discussion and outlook

8.2.1 Redundant tours

In urban areas, only two types of cables are allowed to be used, while targets in rural areas are allowed to connect to any of the three types of cables. The heuristic model in Chapter 4 is now programmed to assign targets to two types of cables, not three, regardless the type of area. Recall that the method is designed for three types of cables.

A large gain is not expected when the program is extended to work for three types of cables: in some rural areas, there are no type 0 or type 1 cables, so a target must be connected to the type 2 cable present there. That means there is no decision to make on which type of cable to connect to. Moreover, the number of street cabinets that are activated in rural areas is usually very low, i.e. one or two, which is not a problem that requires an algorithm.

Also, a large gain is not expected since it is preferred to connect targets to type 0 and type 1 cables as these type of cables are low in the network hierarchy. Connecting to these type of cables instead of type 2 cables yields a more sustainable and efficient network. One way to incorporate this preference into the model is to set penalties on connecting to a type 2 cable. This could be incorporated in the gain function of the heuristic, or in the cost function of the B&B method. However, costs are usually not the only argument for allowing a connection to a type 2 cable. Another possiblity is to compare solutions that have been obtained by allowing and not allowing connections to type 2 cables.

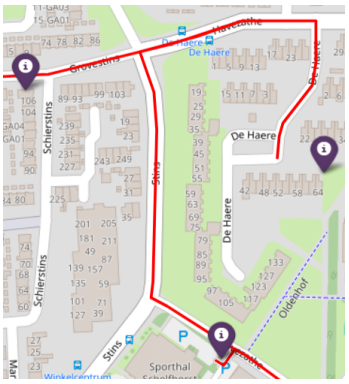


Figure 8.1 – The cables and street cabinets are for illustrative purposes only, the figures do not show the actual locations of cables and street cabinets. The shortest path from the first target (far left) in the tour to the second one (far right) overlaps with the path from the second to the third target (bottom). Therefore, the second target is connected non-redundantly. However, this tour may be the result of the heuristic and exact methods that yield redundant tours.

Another subject of further research is that a tour constructed as a redundant tour may accidentally be a non-redundant tour. A path between nodes was chosen to be the shortest path. When two shortest paths share a part, the tour is partially non-redundant if both paths are in the solution. For an example, see Figure 8.1: the path between the first (far left) and second (far right) target and the second and third (bottom) target overlap. This problem mostly occurs in areas with many dead-end streets or near waterways. This usually only makes a few targets non-redundant, which is allowed as long as the number of customers is below 2000. Therefore, the constructed tour is still feasible even though it is not redundant.

To overcome this problem, all crossroads must be modelled. Instead of using the shortest path between two nodes, the shortest path that does not contain parts that are already in the tour is used. However, this may not yield an optimal tour, and it is not possible to incorporate this in the heuristic method as the new tour depends on the tour constructed in a previous iteration.

Some research has been done on ring-star problems, which are problems in which a subset of the targets are connected in a ring, and the remaining targets are connected non-redundantly to a target or depot that has already been visited. For example, research performed by Baldacci et al. [2] may direct toward a solution, which focusses on network-design problems in telecommunication.

8.2.2 Non-redundant tours

To obtain a non-redundant tour from a redundant one, the model described in Chapter 6 optimises each segment in a tour separately. All targets and (copy) depots in a segment may be rearranged, as long as this action yields a feasible tour. However, all (copy) depots in a segment are of the same type. Consequently, targets cannot switch between cable types, even though that may yield an improvement of the tour. A large gain on the tour cost is not expected by allowing targets in one segment to connect to all types of cables: if a large gain is obtained this way, the target would have been

in its own segment in the redundant tour already. A downside is that the number of feasible paths increases tremendously, which leads to a very slow B&B method.

In Chapter 6, the assumption was made that a target needed exactly one copy target. However, tours may be improved when they include multiple copy targets: consider the example in Figure 8.2. Target V_2 is visited three times, meaning that two copy targets would be necessary and thus is such a tour currently not possible.

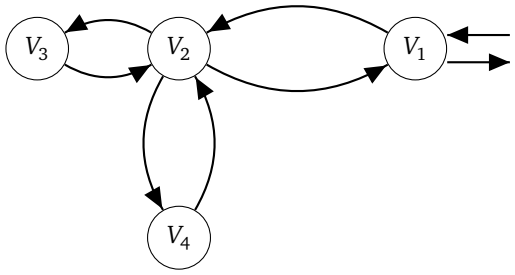


Figure 8.2 – A non-redundant tour that needs at least two copy targets, since target V_2 is visited three times.

The B&B method that was used to obtain non-redundant connections for all segments in a redundant tour finished in acceptable time for all but two problems. The goal is to use the method for larger redundant tours than the ones currently considered. Even though the segments in a tour do not necessarily contain more targets, an improvement in running time is desirable. For example, this can be done by transforming the current algorithm into a heuristic method by setting appropriate pruning rules. Also, the method has been implemented recursively, which is quite slow in Python due to the number of function calls. A solution could be to program the method iteratively or program core parts in a faster language such as C++ or Julia.

To conclude, the goal was to find an optimal tour through a set of targets and (copy) depots for redundant and non-redundant tours. The heuristic method yielded promising results for the redundant tour and its cost. Also, no computational problems are expected when problem sizes increase. The B&B method for non-redundant tours gave insights in the reduction of costs compared to the redundant tours. This method can be improved by reducing the computation time, so that larger problems can be handled.

Bibliography

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem, a computational study*. Princeton University Press, 2006.

[2] R. Baldacci, M. Dell'Amico, and J. S. González. The Capacitated *m*-Ring-Star Problem. *Operations Research*, 55(6):1147–1162, 2007.

[3] CBS. Bevolkingskernen in Nederland, 2011. Centraal Bureau voor de Statistiek, Den Haag/Heerlen, March 2014.

[4] E. Dijkstra. A note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):519–643, 1964.

[6] B. Golden, L. Bodin, T. Doyle, and W. Stewart, Jr. Approximate Traveling Salesman Algorithms. *Operations Research*, 28(3):694–711, 1980.

[7] Y. GuoXing. Transformation of multidepot multisalesmen problem to the standard travelling salesman problem. *European Journal of Operational Research*, 81 (3):557–560, 1995.

[8] M. Held and R. M. Karp. The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18(6):1138–1162, 1970.

[9] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, October 2000.

[10] S. K. Korotky. Semi-Empirical Description and Projections of Internet Traffic Trends Using a Hyperbolic Compound Annual Growth Rate. *Bells Labs Technical Journal*, 18(3):5–21, 2013. doi: 10.1002/bltj.21625.

[11] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[12] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):401–659, April 1973.

[13] V. López and L. Velasco. *Elastic Optical Networks*, chapter 1, pages 1–5. Springer Nature, 2016.

[14] W. Malik, S. Rathinam, and S. Darbha. An approximation algorithm for a symmetric Generalized Multiple Depot, Multiple Travelling Salesman Problem. *Operations Research Letters*, 35(6):747–753, November 2007.

[15] O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the TSP incorporating local search heuristics. *Operations Research Letters*, 4(11):219–224, May 1992.

[16] P. Oberlin, S. Rathinam, and S. Darbha. A transformation for a Multiple Depot, Multiple Traveling Salesman Problem. In *2009 American Control Conference*, pages 2636–2641, June 2009. doi: 10.1109/ACC.2009.5160665.

[17] G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4):376–385, 1991.

Appendix A

List of notation

p	Largest depot type, so $p+1$ is the number of depots
m_i	Number of copy depots of type i
m	Total number of copy depots
n	Total number of targets
V_k	Target, $1 \leq k \leq n$
C_i	Depot, for $0 \leq i \leq p$
$C_i^{(j)}$	Copy depot
r_j	Number of targets that salesman j visits

A.1 Redundant tours

G_T	Transformed graph corresponding to precedence-constrained, asymmetric TSP
$c_T(u,v)$	Cost of the edge (u,v) in the transformed graph G_T
$P_i^{(j)}$	$P_i^{(j)} = (V_{j_1}, V_{j_2}, ..., V_{j_{r_j}}, C_i^{(j)})$
	Tour by salesman j that ends with copy depot $C_i^{(j)}$

A.2 Non-redundant tours

$V_k^{(0)}$	Copy target corresponding to target V_k. Each target has exactly one copy.
$\tilde{G}_T$	Graph corresponding to tours for non-redundant connections
$\tilde{c}_T(u,v)$	Cost of the edge (u,v) in the graph $\tilde{G}_T$
$\tilde{P}_i^{(j)}$	$\tilde{P}_i^{(j)} = (V_{j_1}, V_{j_2}, ..., V_{j_{r_j}}, V_{j_{r_j}}^{(0)}, ..., V_{j_2}^{(0)}, V_{j_1}^{(0)}, C_i^{(j)})$
	Tour by salesman j that ends with copy depot $C_i^{(j)}$.
$\tilde{p}_i^{(k)}$	A segment in $\tilde{P}_i^{(j)}$ that contains $\tilde{r}_k$ segments.
$\tilde{r}_k$	Number of targets in $\tilde{p}_i^{(k)}$
$\tilde{m}_j$	Number of copy depots in $\tilde{P}_i^{(j)}$

A.3 Precedence-constraints

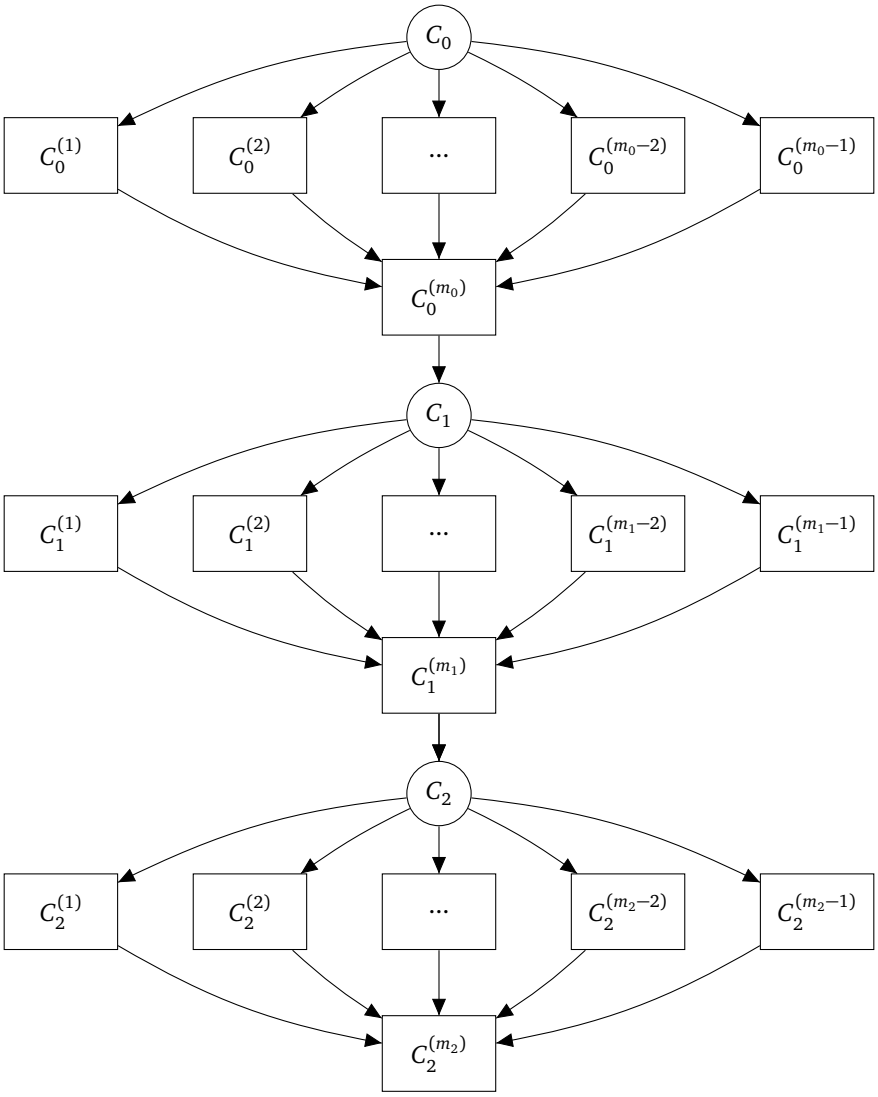


Figure A.1 – This diagram shows the precedence constraints for the depots and copy depots. An arrow gives the ordering of the visits of the (copy) depots. All tours start with depot C_0, from which all copy depots of type 0 but the last one are reachable. Before the last copy depot of type 0 can be visited, all other copy depots of type 0 should have been visited. From $C_0^{(m_0)}$, only depot C_1 can be reached. The same holds for copy depots of type 1 and 2.