Master thesis

Master mathematics

Analyzing variations of the Vehicle Routing Problem

Author:
Erik Bosch
4073460

Supervisor:
Leen Stougie

July 17, 2017

Radboud University

Contents

1 Introduction

This thesis will analyze variations of a specific problem that is contained in combinatorial optimization, the Vehicle Routing Problem. The problem takes a look at a depot and a number of cities. The depot has a fixed number of trucks and has to visit every city to deliver a specific amount of supplies. The supplies that every city needs might be different and the supplies that every truck may hold might also vary per truck. The original problem is to find a set of tours from the depot to a number of cities and back to the depot in a way that the trucks can actually deliver the supplies that are needed to each city on this tour. [4]

This problem can be altered in different ways. One could increase or decrease the number of trucks, the number of depots, change the restrictions on what cities need or create other variations. In general, the Vehicle Routing Problem is not easy to solve and is even NP-hard [5]. A lot of research on this problem and its many variations is already done and most of it will be irrelevant for this thesis. If one would like to read more on Vehicle Routing Problems then the articles written by M. Baker and K. Sundar are recommended [2][3].

The main focus of this thesis is to analyze specific variations of the Vehicle Routing Problem. The problems will be visualized by looking at a mechanical arm that has to construct a chip by adding components to it. The mechanical arm is restricted by worldly problems, but the specific problems can be seen as Vehicle Routing Problems with specific restrictions. For example, the number of "trucks"that are available is infinite to visualize that the mechanical arm can be reused every time that it visits a reservoir. More specific definitions will be given in the next chapter.

Throughout this thesis, the Edmonds' Blossom Algorithm will be extensively used. More specifically, the Blossom V algorithm will be used. This is an extension of the original Edmonds' Blossom algorithm and it runs in less time than the original. The running time of Blossom V is $O(|N|^3|M|)$, where N is the set of nodes of a given graph and M is the set of edges. This thesis will always use a complete graph and the cost will therefore always result in $O(|N|^5)$.
Further details on the Blossom V algorithm are not required to fully understand the remainder of this thesis. If one would wish to read more details, then they can be found in the article written by V. Kolmogorov [1].

2 General definitions

The first section will be dedicated to establishing some basic rules about the problems and constructing a basis to work from. The intuition that is used is that of a mechanical arm, which needs to construct a chip by adding components to it one at a time. The mechanical arm is restricted in some ways. First, it is limited by the amount of components that it can hold at a time. It has to return to some kind of reservoir to renew its stack of components if it runs out at any time. Furthermore, the amount of these reservoirs is also limited and the location has to be fixed at the start. The reservoir can therefore not tag along with the arm to supply it on the go. Lastly, the arm might be limited in its movement options. Some mechanical arms will not be able to move in every possible direction. This all has to be reflected in some mathematical model.

As first the chip itself will be defined. This will be done by looking at the rectangle of the plane $[0, L] \times [0, H]$. This rectangle will be used to determine all locations that need to be visited by the arm. These locations will be called customers and will be denoted by $n, m, k, l, n_1, \ldots, n_k$ and so on. The set of all customers will be denoted N. The reservoirs will be denoted by $r_1, \ldots, r_k$ and so on and the set of all reservoirs will be denoted R. Once in a while, the notation will be abused slightly and it will be denoted that $|R| = R$. The context will make clear which interpretation should be used. In the problems that will be analyzed the reservoirs can only be located on the line x-axis. It will be assumed that all reservoirs are numbered from left to right depending on their location on the axis. Furthermore it was already assumed that the mechanical arm can only take a set amount of components with it at a time. The maximum number of components that the arm can hold will be denoted by C.

It is now possible to give a description of the problems that will be analyzed.

P_R^C is the problem in which the location of the reservoirs is not set and has to be chosen somewhere on the x-axis. The movement of the arm is not limited by any means. Furthermore C denotes the maximal amount of components that the arm can hold and R denotes the number of reservoirs that should be placed.

In $\overline{P_R^C}$ the location of the reservoirs is fixed somewhere on the x-axis and can not be chosen. This is the only difference with P_R^C.

In D_R^C the movement of the arm is restricted and it can only move left, right, up or down. Furthermore the problem is the same as P_R^C.

The goal in all these problems is to find the cheapest path, that adds a component to every customer. To solve this, the problems will be imagined as a weighted directed graph $G = (V, E)$ in which V is the set of all customers N together with all reservoirs R, $V = N \cup R$. E will simply be every possible directed edge between all nodes V. The weight of an edge (v, w) is the shortest distance that the mechanical arm has to travel if he starts in v and ends in w. This means that the Euclidean distances will be used mostly. When analyzing D_R^C, the L_1 metric will be used instead. Furthermore, the graph is directed, but this will be used loosely throughout the thesis. The only reason that the graph is being imagined as directed is because in some cases one might have to walk the same edge two times in two directions. This is called a single tour and will be discussed later on. When the graph is not directed, this might cause ambiguity. Lastly, walking from a customer v to itself is a useless action. Therefore all edges from one node to itself will be removed.

A feasible solution will then be a path that starts in one reservoir and walks through all customers in N, while simultaneously walking back to reservoirs, if it has to restock on components. One could also see a feasible solution as a collection of tours from one reservoir to another reservoir, while in between visiting customers only. Both these views will be used in the coming chapters. The path described will be called P or P^* when speaking about an optimal path. In some cases it will be seen as a path, as was described first and when it is convenient it will be seen as a set of tours, which a path of this form will essentially always be. Also, one could see P as an ordered set of directed edges and this view will also be used throughout the thesis.

An optimal solution will then be the cheapest feasible solution. One should notice that in an optimal solution, the tours between reservoirs never have to visit more than C customers. It is therefore assumed that every tour visits at most C customers.

Furthermore, something has to be said about the way that proofs will be handled throughout this thesis. One should notice that it is possible to reverse the direction of every tour. This is true, because it does not matter where a tour starts, because it will always start and end in some reservoir. An important thing to notice is that it is possible to go from one reservoir to another reservoir through a tour. These special tours will be defined more accurately later on. One should notice that the direction of these tours can also be reversed. Furthermore, one should notice that if there are enough of these tours between reservoirs, that it is always possible to construct a feasible solution. This method will be used a lot in different proofs and the reader should check for himself that this is indeed true.

Throughout this thesis different ways of proving will be used. The most common one will be to interchange edges from different tours to create new tours, or to interchange tours all together. One should always check, with the above statement, that it is truly possible to construct a new feasible solution when these methods are used. In most cases, this will not be difficult, but in some cases one might need to draw a picture to make it more clear. If proofs are truly difficult, a picture will be provided that hopefully will enlighten the proof.

At the end of this thesis another problem will be analyzed. This will be a problem in which not all components are equal anymore. There will be different kinds of components and customers will need a specific kind. A lot that was said about the other problems will not be true for this problem. This chapter will therefore start from scratch without any of the assumptions stated above.

3 Fixed reservoirs, Euclidean distances

This chapter will take a closer look at the problem $\overline{P_R^C}$ for different values of C and R. The previous chapter discussed the Vehicle Routing Problem and it was noted that $\overline{P_R^C}$ is very similar to this. The main difference will be that $\overline{P_R^C}$ will always take place in a 2-dimensional grid, where the Vehicle Routing Problem does not have this limitation. The articles mentioned in the introduction take a closer look at the problem in more dimensions, but this will not be the focus of this thesis.

This chapter will solve $\overline{P_R^C}$ to optimality in polynomial time for some values of C and R. During the chapter an indication will be given on when the problem starts to be more difficult. The starting point will be set at $C = 2$ and $R = 1$, since this is the most simple case that is not trivial. After that R will be incremented and at the end there will be a mention of what happens when C is incremented instead.

3.1 One reservoir, two components

The following chapter is dedicated to $\overline{P_1^2}$. First a few lemmas will be proven that will turn out to be useful. Then an algorithm will be constructed and finally it will be proven that this algorithm solves this instance of $\overline{P_R^C}$ to optimality. Also the running time will be given and it will turn out that this instance can be solved to optimality in polynomial time.

3.1.1 Preliminary results

The following section will be used to set up some lemmas. The first lemma will note that in an optimal solution, it is never optimal to visit the same customer twice.

Lemma 1. *There exists an optimal solution in which, for every point $v \in N$ there exists exactly one $w \in V$ such that $(w, v) \in P^*$.*

Proof. It is assumed that v is a customer and that a feasible solution starts and ends in a reservoir. The fact that there is at least one such w becomes therefore trivial, since a feasible solution has to visit every customer at least once. This implies that there exist at least one edge going to v.

Assume now that there is an optimal solution in which there exists a $w' \neq w$ such that there exists another edge (w', v) that is used in the optimal solution. It is now possible for this tour to either visit three customers, but not use a component at v, or that it visits v and then returns to the base. In both cases one should easily notice that it is cheaper to go directly to either the third customer or to base. This is only true in a graph in which triangle inequality holds, but since the graph in question is induced by Euclidean lengths, this will not be a problem. It can now be concluded that the lemma holds. □

From now on it will be assumed that any optimal solution visits every customer exactly once. Furthermore a corollary follows from this lemma.

Corollary 1. *If in the optimal solution there exist $v \in N$ and $r \in R$ with $(r, v), (v, r) \in P^*$. Then there does not exist any $w \in N$ such that (w, v) is used in the optimal solution.*

Proof. The proof to this corollary is trivial. □

This corollary leads to the definition of a single tour, which will be given now.

Definition 1 (Single tours)**.** *If there exists a $v \in N$ and $r \in R$ such that $(r, v), (v, r) \in P$, then these two edges will be called a single tour.*

A single tour might look really inefficient, since triangle inequality assures that visiting another customer in the same tour is always cheaper then visiting them one at a time. However, in some cases it is necessary to use single tours, for example when the number of customers is odd. Though, using too many single tours is inefficient as will be proven in the next lemma.

Lemma 2. *For the problem $\overline{P_1^2}$ there exists an optimal solution with at most one single tour.*

Proof. Assume that there is an optimal solution with at least two single tours. Assume that these single tours are connected to customer n_i and n_j. The first lemma of this chapter assures that these customers are not equal to each other. Construct a new solution by replacing the single tours by one tour that goes from r_1 to n_i to n_j to r_1. This is surely not more expensive than the solution that was presented before and therefore it must be optimal. This can be done with all pairs of single tours to receive an optimal solution with at most one single tour. It must be noted however that it might be possible that the optimal path has to be altered somewhat, but this is easily done.

Also, one should note that there exist examples in which there exist optimal solutions with more than one single tour. These solutions can then be altered in the same way to receive an optimal solution with at most one single tour, but these solutions can both be optimal. □

The above lemmas give a clear indication on how an optimal solution should look like. If there is an even number of customers, there will be no single tours in an optimal solution. This is clear because of lemma 2. An odd number of customers will give exactly one single tour. This means that an algorithm must match all the remaining points to optimality, because a tour of three customers is not allowed and more tours of one customer are not allowed either. This will be done by using the Edmond Blossom V algorithm, which was shortly described in the introduction.

3.1.2 Constructing the algorithm

This section will give the pseudocode of an algorithm that solves this instance of the problem to optimality in polynomial time. The optimality and running time will be proven in the section after the algorithm.

This algorithm constructs an ordered set P, which will turn out to be an optimal path.

Input: A set N of customers and a location for r_1
initialize: An empty ordered set P

Determine if $|N|$ is odd or even

if $|N|$ even **then**
    Use Blossom V to find the optimal matching M of N
    Number edges $e_1, \ldots, e_{\frac{|N|}{2}}$ in M
    for $i = 1, \ldots, \frac{|N|}{2}$ **do**
        Add $(r, v_i), (v_i, w_i), (w_i, r)$ to P, where $e_i = (v_i, w_i)$
    end for

else if $|N|$ is odd **then**
    for all $n \in N$ **do**
        $P = \emptyset$
        Use n as a single tour
        Blossom V: Find optimal matching M of $N - \{n\}$
        for $i = 1, \ldots, \frac{|N|}{2}$ **do**
            Add $(r, v_i), (v_i, w_i), (w_i, r)$ to P, where $e_i = (v_i, w_i)$
        end for
        Add $(r, n), (n, r)$ at the end of P
        Check if the current solution is cheaper than previous cheapest solution and save it
    end for
end if
Output: An ordered set P, which will be an optimal path

3.1.3 Proving optimality

First it will be proven that the algorithm indeed finds an optimal solution. If there is an even number of customers, it is known that the optimal solution can have no single tours. Since every customer has at least one edge connecting it to the reservoir, the only part that can be optimized is the matching. Indeed, the Blossom V algorithm finds the optimal matching for these customers. In case that the number of customers is even, the algorithm is therefore optimal. If there is an odd number of customers the lemmas show that there is exactly one single tour. The algorithm tries every customer for that single tour and computes the optimal solution for that single tour customer. The cheapest one of these $|N|$ solutions is therefore the optimal solution. In both cases the algorithm will eventually find the optimal solution and therefore it can be concluded that the algorithm is correct.

Now lets consider the running time of the algorithm. In case of an even number of customers it is dominated entirely by the time it takes to find an optimal matching. This is done by using the Blossom V algorithm, which runs in time $O(|N|^5)$. If the number of customers is odd, then the algorithm will need to do this $|N|$ times, because it will use every customer as a single tour exactly once. This means that it runs in time $O(|N|^6)$. The remaining steps of the algorithm do not require more time than this. The algorithm needs to sort the customers and sometimes sort edges, but this can be done in less time. Therefore the running time of this algorithm in the worst case scenario is $O(|N|^6)$ which is polynomial and therefore the specific problem is part of P.

3.2 Two reservoirs, two components

The next section will expand the problem of the last section by adding another reservoir. This is very similar to the last problem, but already much more complicated. A few new lemmas need to introduced, as will be done in the first section. After that another algorithm will be constructed and again it will be proven that this algorithm is optimal and that it runs in polynomial time. This will mean that $\overline{P_2^2}$ is also in P.

3.2.1 Preliminary results

Since there are now two reservoirs, it would be convenient to know which customer is closest to which reservoir. This will be done in the following definition.

Definition 2. *The set N_{r_i} is the subset of N such that for every $n \in N$ if $d(r_i, n) \leq d(r_j, n)\ \forall j$, then $n \in N_{r_i}$, where $d(\ldots, \ldots)$ is the distance function.*

This definition is very general and can be used in a multiple of scenarios. Later on, this definition will be used with a different metric and also with more reservoirs. The subset N_{r_i} will sometimes be called the neighborhood of r_i.

Furthermore, it must be noted that lemma 1 and corollary 1 still hold for this specific problem. The proof has to be altered slightly, but this is not hard and can be checked easily by the reader. Corollary 1 can also be extended to be more general in the case of several reservoirs.

Corollary 2. *If there is an optimal solution and there exist $r_i, r_j \in R$ and $v \in N$ such that $(r_i, v), (v, r_j) \in P^*$, then there does not exist a $w \in N$ such that $(w, v) \in P^*$.*

Proof. The proof for this corollary is again trivial. □

This gives a motivation for extending the notion of a single tour.

Definition 3. *A single tour is defined as a tour $(r_i, v), (v, r_j)$, for a customer $v \in N$ and reservoirs $r_i, r_j \in R$, where r_i and r_j can be the same.*

Since there are now more reservoirs, one could expect that there exists a situation in which the path P travels directly between two reservoirs, without visiting any of the customers. In the next lemma it is proven that there is always at least one optimal solution in which this does not happen.

Lemma 3. *There exists an optimal solution in which P does not traverse an edge (r_i, r_j) where $r_i, r_j \in R$ and $i \neq j$.*

Proof. Assume that there is an optimal solution such that the edge (r_i, r_j) is traversed in P^*, such that $r_i, r_j \in R$ and $i \neq j$. Without loss of generality, assume that $i = 1$ and $j = 2$. First note that there is one trivial case, namely when there are no customers. In this case an empty solution is optimal and feasible. Another trivial case happens when either r_1 or r_2 is not being used at all. In this case, the extra edge will be completely useless and can be removed without any trouble. From now on it is therefore assumed that both reservoirs are being used in the optimal solution.

There are now two possible scenarios left. Either all tours start and end in the same reservoir or there exist at least one other tour that starts in one reservoir and ends in the other.

In the first case it is known that there is at least one tour connected to r_1. Assume that this tour goes from r_1 to m to n and back to r_1. It is clear that the edge (n, r_2) is not more expensive than the edges $(n, r_1), (r_1, r_2)$ due to triangle inequality. Replacing these edges and altering the optimal path will result in a feasible solution that is not more expensive and therefore an optimal solution. In this solution the edge (r_1, r_2) is removed and the lemma therefore holds. It must be noted that the tour connected to r_1 could be a single tour, but this will not matter for the proof and the result will still hold.

In the second case one should note that without (r_1, r_2) there are already enough ways to travel from r_1 to r_2. Since (r_1, r_2) does not service any customers, it can be removed and it will still be possible to construct a feasible solution from the remaining edges. This solution is clearly cheaper than the previous, which means that this contradicts the optimality of the previous solution.

In all cases it is possible to remove the edge (r_1, r_2) and therefore the lemma must hold. □

Also, one might wonder if all the new reservoirs are really necessary or that some can be ignored. The next lemma will prove that in special cases, one of the reservoirs is not needed. These are special cases, but the algorithm at the end of this section will use these special cases to sometimes be a little more efficient.

Lemma 4. *If there exists an i such that $N_{r_i} = N$, then there exists an optimal solution that only uses reservoir r_i.*

Proof. Assume without loss of generality that $N_{r_1} = N$ and that there is an optimal solution in which there exists an edge (v, r_2) or (r_2, v). Replace all these edges by (v, r_1) and (r_1, v) respectively. This will surely not be more expensive than the original solution. It is now easy to see that this creates a situation in which there are only tours starting and ending in r_1 and it is easy to find a feasible path. Therefore this new situation is again a feasible solution and so it must be an optimal solution. In this solution r_2 was not used at all and therefore the lemma must hold. □

In lemma 3 a lot of tours were used that started in one reservoir and ended in the other. When using two or more reservoirs, it becomes a necessity to use these tours. The next lemma will show that there is always such a tour when not all points are close to one reservoir.

Lemma 5. *If for all $i = 1, 2$ $N_{r_i} \neq N$, then for every optimal solution there exist $v, w \in N$ such that $(r_i, v), (v, w), (w, r_j)$ are traversed or there exists $v \in N$ such that $(r_i, v), (v, r_j)$ are traversed, where $i \neq j$.*

Proof. Assume now that there is an optimal solution in which the lemma does not hold. Then only one reservoir is used. Without loss of generality, assume that this reservoir is r_1. Since $N_{r_1} \neq N$ there exists a $v \in N$ such that $v \in N_{r_2}$ and $v \notin N_{r_1}$. Now replace the tour of customer v by a tour from r_2 to v to w to r_1. If v was a single tour then this w does not exist, but this does not matter and it can just be replaced by a tour from r_2 to v to r_1. Altering the path a little bit will again give a feasible solution. Since $v \notin N_{r_1}$ this new solution is cheaper than the previous solution. However, it was assumed that the previous solution was optimal, which gives a contradiction. This must mean that the assumption was wrong and that the lemma must hold. □

This lemma implies that in general an optimal solution will need more than one reservoir. This can only be done if there is a way to travel from one reservoir to the other. Lemma 3 assures that

this will not be done directly but via a tour going from one reservoir to the other and visiting customers in between. This gives rise to the next definition of a switch.

Definition 4. *A switch is a series of edges* $(r_i, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)(v_k, r_j)$*, where* $v_l \in N\ \forall l \leq k$*, with* $k \leq C$ *and* $i \neq j$*.*

The given definition is more general and will be used in future sections too. For this chapter, C is equal to two and a switch will either consist of one or two customers. A switch with only one customer will be called a single tour switch from now on.

This brings attention to the fact that single tours will still exist, even when using two reservoirs. The next lemmas will prove a few characteristics of single tours when there are at least two reservoirs.

Lemma 6. *There exists an optimal solution such that for all* i *there exists at most one point* $v \in N$ *such that* $(r_i, v), (v, r_i)$ *are traversed in the solution.*

Proof. Assume that there is an optimal solution in which this does not hold. Then there must exist $v, w \in N$ such that they are both connected to r_i as a single tour. However, this situation is the same as in lemma 2 and the same proof can be applied to this lemma. □

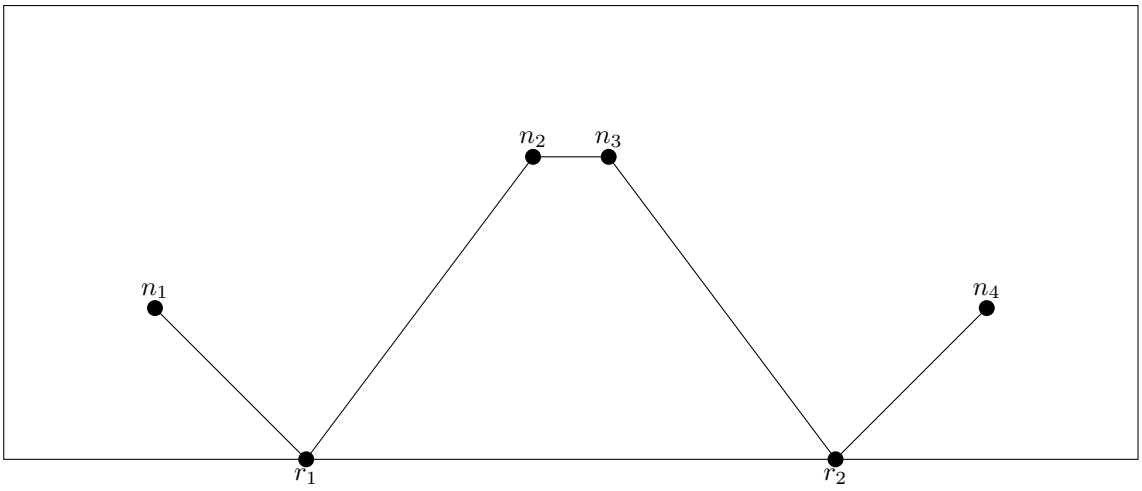
This lemma implies that every reservoir will have at most one single tour, which is similar to the previous section. With more reservoirs however, there is the possibility that there exists a single tour between two reservoirs, which was called a single tour switch. When this is the case, there will still be only one single tour connected to each reservoir, which will be the single tour switch between the two.

Lemma 7. *There exists an optimal solution such that if there exists a single tour switch between* r_1 *and* r_2*, then there are no other single tours connected to either of them.*

Proof. Assume that there is an optimal solution with a single tour switch that goes through customer n and assume that there is also a single tour attached to either one of the reservoirs. Without loss of generality, assume it is connected to r_1 and that it goes through customer m. Triangle inequality then assures that the edge between r_1 and n and between r_1 and m are surely not cheaper than the edge between n and m. Replacing aforesaid edges by the latter one will result in a collection of edges that is still optimal. By altering the optimal path slightly, a new feasible solution can be made with these edges. This new solution will be optimal and this proves the lemma. □

This lemma implies that if there is an even amount of customers, there will never be a single tour switch. There is still the possibility of two single tours, each connected to a different reservoir.

Example 1. *The next example shows a situation in which there are two reservoirs and four customers. The optimal solution in this example will be a solution which has two single tours. This means that the algorithm now has to account for single tours, even when there is an even number of customers. This is different from the last section, but will not result in much trouble.*



Lemma 8. *If there is an optimal solution and there exists a v such that if v is a single tour connected to r_i, then $v \in N_{r_i}$.*

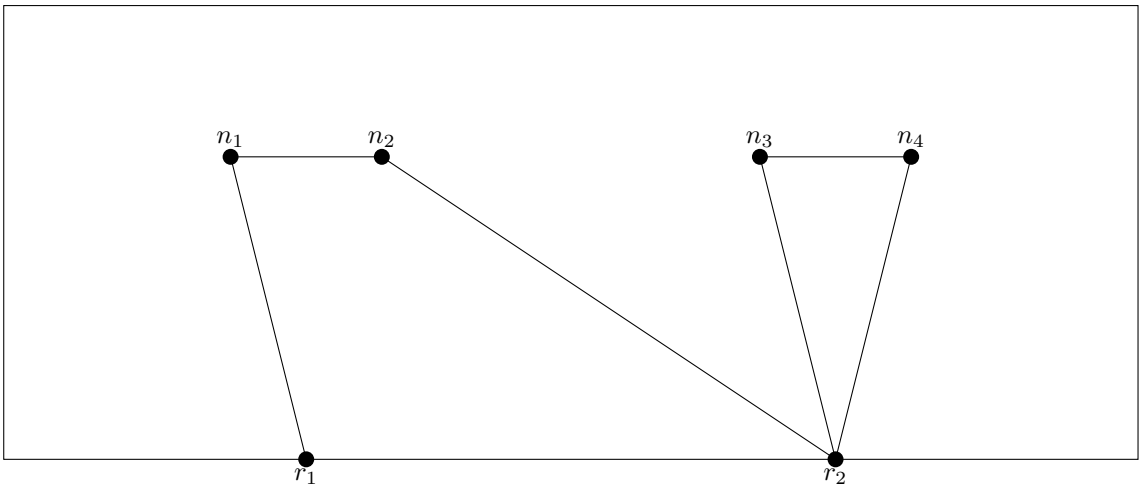
Proof. The proof of this is trivial. □

This gives a clearer view on what an optimal solution should look like. If there is an even number of customers, most of the time the solution will need a switch between the reservoirs. This will never be a single tour switch, but it is possible that there are two single tours, each connected to different reservoirs. These single tours will be connected to the reservoir closest to them.

If the number of customers is odd, then there is a maximum of one single tour. This might be a single tour switch or it might be connected to one of the reservoirs. This gives a clear view on what restrictions lie on the possible solutions. The remaining solutions can all be checked and this does require only polynomial time as will be seen later on.

A few more remarks must be made. Intuitively one might think that all switches will go from a customer close to one reservoir to a customer close to the other reservoir. This might sound optimal, because connecting customers to their closest reservoir is most of the time a good idea to reduce the cost. However, this does not necessarily have to be true and the next example will show a counterexample of this thought.

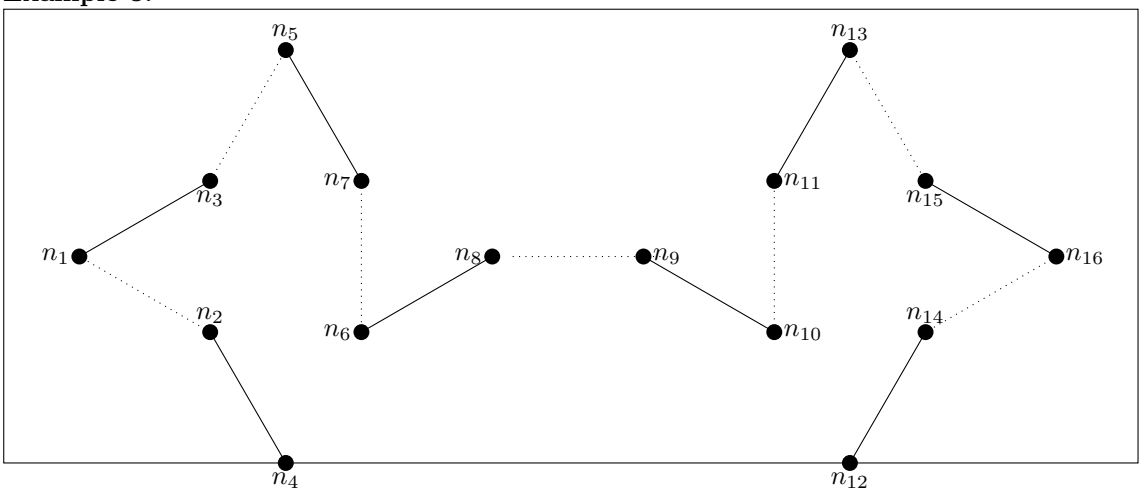
Example 2. *In the next example n_2 is clearly closest to r_1. However, in this solution it is always cheaper to pick a switch like below instead of switch from, for example, n_2 to n_3. This is mostly because in that case n_1 and n_4 would be connected to each other or would be single tours. Both scenarios increase the cost massively.*



This example implies that when an algorithm tries to check all possible switches, it shouldn't just pick a pair where one customer is close to one reservoir and the other close to the other. It should consider all possible pairs of customers as potential switches.

A few remarks have to be made about optimizing this problem. One might expect that the optimal matching of the customers should always be used in the optimal solution. Sadly, this is not true and the following example will give a counterexample to this.

Example 3.

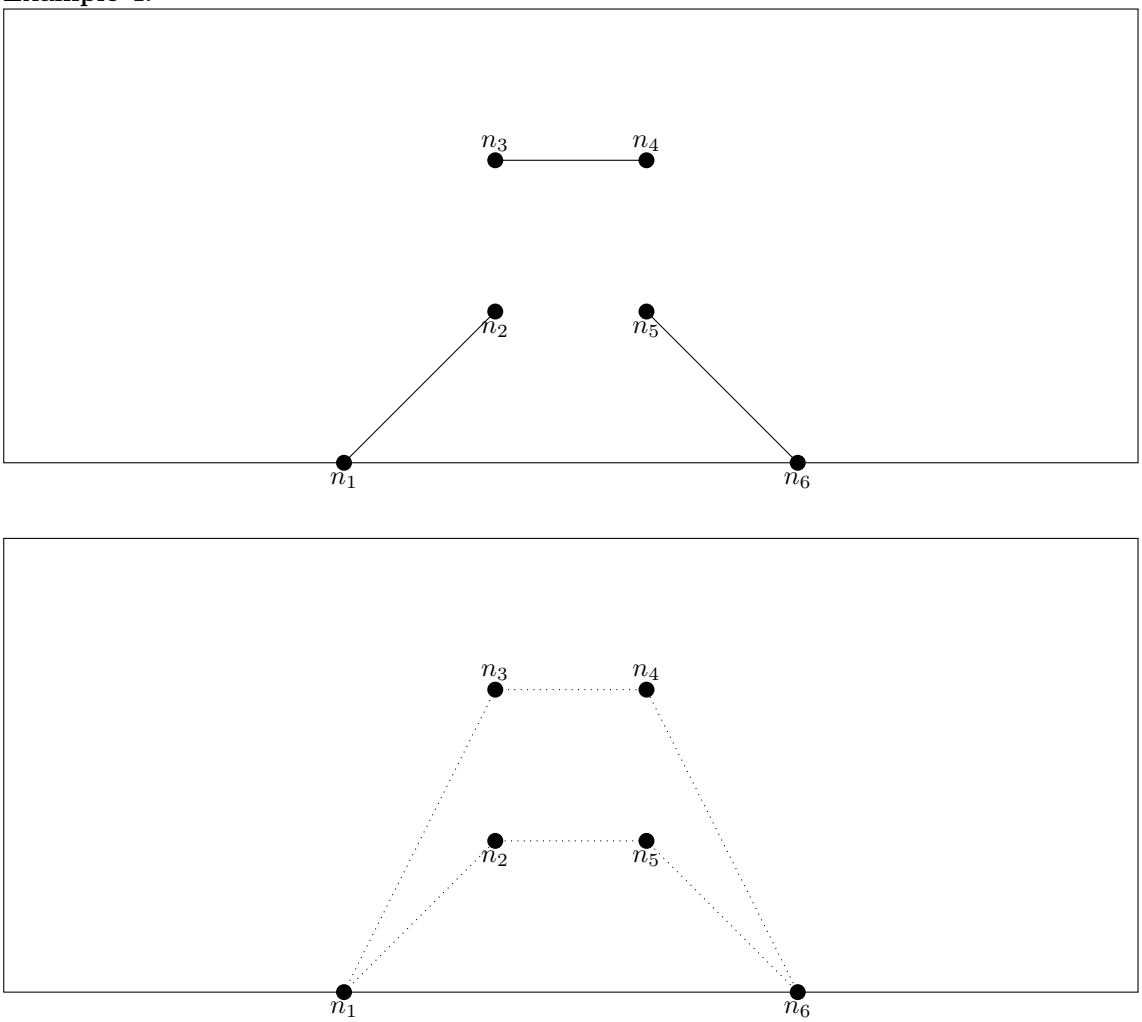


In this example the customers are chosen to be at least 2 away from each other, The fat lines are an optimal matching, but setting n_4 and n_{12} to be single tours and using the dotted lines as matching will result in a feasible and cheaper solution.
One might think that this is a shady example, because n_4 and n_{12} lie both directly on the reservoir. However, by solving a few equations, one will find out that moving n_4 and n_{12} slightly up will still give the same situation that was sketched before. When the points are on the line, the example is more intuitively though.

Some readers might have noticed that the optimal matching in the previous example did not have a switch and started wondering if this was not crucial. However, the next example will

show that even when there is a switch, then the optimal matching. does not have to be used in the optimal solution.

Example 4.



In this example customer n_1 lies exactly at a distance of 3 from n_6. Both n_2 and n_5 lie at a height of 1 and the other two customers lie at a height of 2. n_2 and n_5 lie a distance of 1 from each other, just like n_3 and n_4.

In the first example, the optimal matching is drawn. If this matching was used for the optimal solution, one would connect all customers to the closest reservoir and this would mean that the distance between r_1 and n_2 would be traversed twice, just like the distance between n_5 and r_2. This would result in a total cost of $4\sqrt{2} + 2\sqrt{5} + 1$.
In the second example both n_1 and n_6 have turned into single tours and the dotted line indicates the new solution that is used. In this case the total distance becomes $2\sqrt{2} + 2\sqrt{5} + 2$, which is cheaper then the cost found before.

This means that one can never be sure that the optimal matching of the customers will be used

for the optimal solution or not. One might again think that this example is shady for the exact same reason as the last example. Again, solving some equations will reveal that both customer n_1 and n_6 can be moved slightly up and the same situation will occur.

3.2.2 Constructing the algorithm

The attempts to cheat a little bit on the running time by just using the optimal matching are in vain. However, the lemmas have proven enough to construct a polynomial time running algorithm. This algorithm will first determine if both reservoirs have to be used. If one is sufficient, it will use the algorithm from the previous section. If both reservoirs have to be used, then it will try every pair of customers to be a switch, together with all possible amounts of single tours. Together this checks every solution that could potentially be an optimal solution.

Just as in the previous section, the algorithm will construct an ordered set P, that will turn out to be an optimal path.

Input: A set N of customers and a location for r_1 and r_2
Initialize: An empty ordered set P

Determine if $|N|$ is odd or even
Determine N_{r_1} and N_{r_2}
if There exists i such that $N_{r_i} = N$ **then**
    Use the previous Algorithm
else
    if $|N|$ even **then**
        for all $n.m \in N$, $n \neq m$ **do** ▷ The switch
            for all $a \in N_{r_1} - \{n, m\}$ **do** ▷ The single tour of r_1
                for all $b \in N_{r_2} - \{n, m\}$ **do** ▷ The single tour of r_2
                    $P = \emptyset$
                    Set n, m as a switch and a, b as single tours
                    Using Blossom V to find the optimal solution
                    Check if it is cheaper than the cheapest solution yet
                    Save it if it is
                    Find the optimal solution with n, m as a switch without single tours
                    Save it if it is cheaper
                end for
            end for
        end for
    else if $|N|$ is odd **then**
        for all $n.m \in N$, $n \neq m$ **do** ▷ The switch
            for all $a \in N - \{n, m\}$ **do** ▷ The single tour or single tour switch
                $P = \emptyset$
                Set n, m as a switch and a as single tour to its closest reservoir
                Using Blossom V to find the optimal solution
                Check if it is cheaper than the cheapest solution yet and save it if it is
                Find the optimal solution with a a single tour switch and do the same as above
            end for
        end for
    end if
end if
Output: An ordered set P, which will be an optimal path

3.2.3 Proving optimality

Thanks to the lemmas proven in this section, it is known that the optimal solution has to be of a certain form. Either it will have to use only one reservoir, in which case the above algorithm will refer to the previous algorithm given, which was already proven to be optimal and polynomial. If both reservoirs are being used, then there must be a switch. The algorithm will try all possible switches and find the optimal solution for all these switches. This means that at some point the algorithm will find the optimal solution and therefore the cheapest of all these solutions must be an optimal solution. It is possible that the optimal solution uses single tours. The algorithm tries all possible combinations of single tours and therefore at one point, if the optimal solution uses single tours, it will come across this solution too. To summarize, the algorithm works, because it tries every potentially optimal solution and therefore one of them has to be the real optimal solution.

The algorithm tries all possible solutions and there are quite a lot. If $|N|$ is even there are $O(|N|^2)$ possible switches, by just considering every pair of customers. Also there are $O(|N|^2)$ possible single tours, which gives a total of $O(|N|^4)$ possible solutions with single tours. There are also possible optimal solutions without single tours, but the amount of solutions with single tours dominates this number entirely. The algorithm will need to find an optimal matching for all these solutions, which can be done in $O(|N|^5)$ time with Blossom V. This means that if there is an even number of customers, the running time is $O(|N|^9)$.

If there is an odd number of customers, there are still $O(|N|^2)$ possible switches. However, there are only $O(|N|)$ possible single tours. This gives a total of $O(|N|^3$ possible solutions. Again, for every possible solution the algorithm needs $O(|N|^5)$ time to find an optimal matching, which gives a total time constraint of $O(|N|^8)$. The algorithm also need to check all possible single tour switches, but this adds only a time of $O(|N|^6)$, which is dominated already by the previous running time.

In the worst case scenario the algorithm needs $O(|N|^9)$ time to solve the problem to optimality. This is still polynomial and therefore the problem finds itself in P.

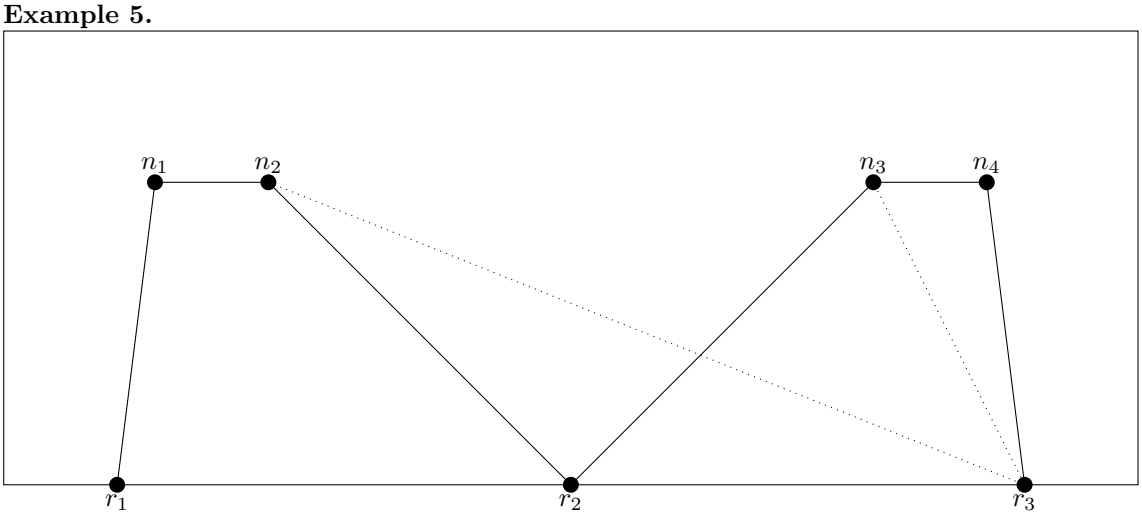
3.3 Three reservoirs, two components

This section will expand even further by adding another reservoir. This might seem like a change that does not add much, but the problem will get even more complicated than before. The first section will therefore be dedicated to setting up more lemmas and refining earlier ones to the new situation. An algorithm will again be constructed and it will be proven that this algorithm solves the problem to optimality. A running time will also be given.

3.3.1 Preliminary results

First of all, it would be wise to check if any of the reservoirs is not needed. Lemma 4 still holds with three reservoirs and will in general hold for any fixed amount of reservoirs. The proof of the extension is the same as the proof of lemma 4. However, one would also think that if there are two reservoirs with all customers in their neighborhood, that the third reservoir is not necessary used in an optimal solution. This is true, but only when the third reservoir is not in the middle. In some cases it might be optimal to use the middle reservoir as some kind of in between stop. An example will be shown next.

Example 5.



When r_2 would not be used, then the optimal solution would go from n_2 to r_3 to n_3. However, when r_2 is used, n_2 can go to r_2 and from there go to n_3. When the distance between r_2 and r_3 is equal to 3 and the distance between n_2 and n_3 is equal to 4 and the whole problem is symmetrical around r_2, then the first path from n_2 to n_3 would be $\sqrt{29} + \sqrt{5}$. The second path would then become $2\sqrt{8} = \sqrt{24}$ which is clearly cheaper.
This example show that even though r_2 has no customers close to him, it is still optimal to use the reservoir.

This shows that one should take caution when ignoring reservoirs that have nothing close to them. In this example the problem that occurs is that r_2 is the middle reservoir. It can therefore still be used as a pit stop to lower the cost of switches. In the next lemma it is shown that this only happens when the third reservoir is in the middle.

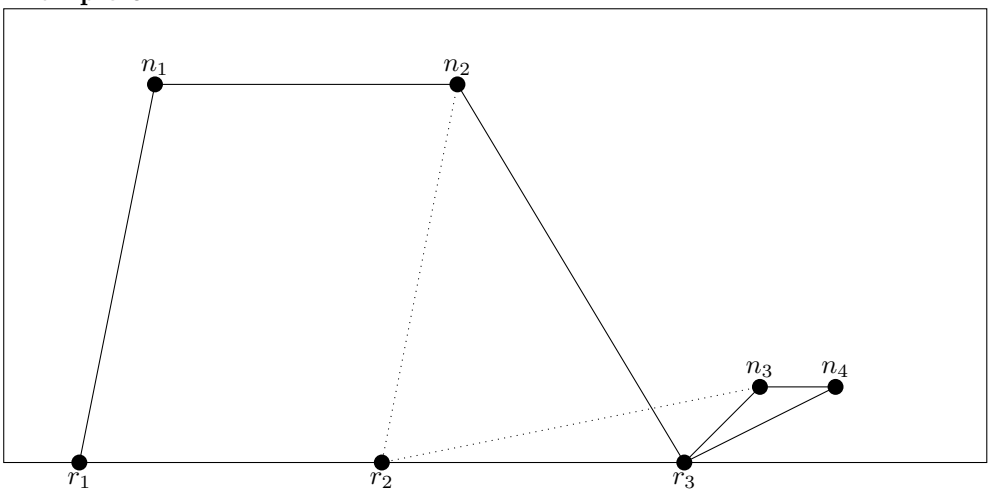
Lemma 9. *If there exist i and $k \geq 0$ such that $N_{r_i} \cup \cdots \cup N_{r_{i+k}} = N$ then there exists an optimal solution that uses only reservoir i to $i + k$.*

Proof. The proof is really straightforward. Assume that this situation occurs and assume that there is another reservoir that is being used. Then one can remove all these edges and replace

them by edges to the closest reservoir. Since this new reservoir must lie on the outside, this won't mess up any switches and the new solution will surely not be any more expensive. It is therefore again optimal and the lemma holds. □

One should wonder what has to happen when the third reservoir is the middle reservoir. At first glance one might think that the reservoir will always be used in some way, even without some customers in its neighborhood, just like in the example above. However, this is not always the case, as is shown below.

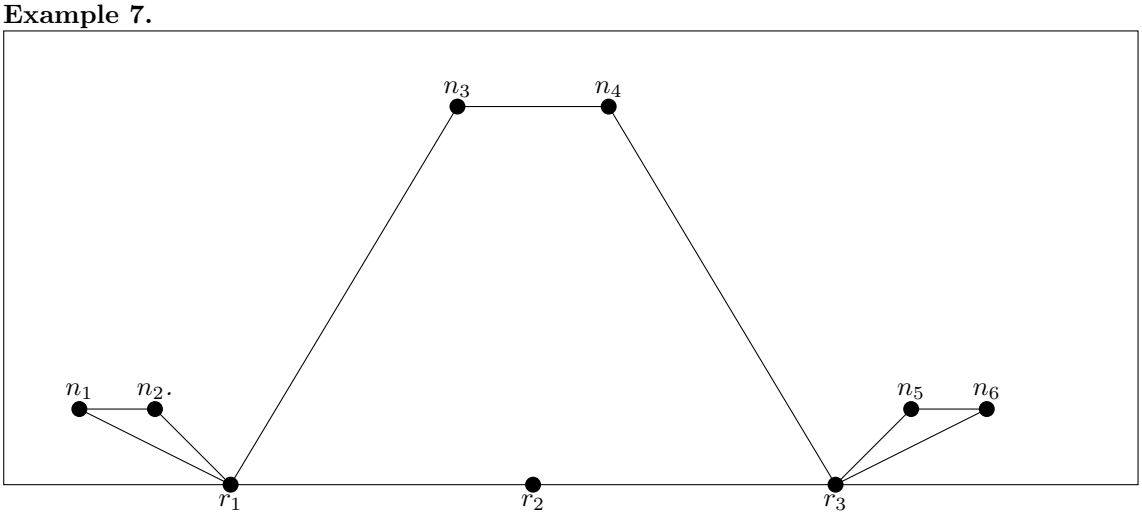
Example 6.



The above example doesn't need much explanation. It is cheaper to use n_2 as a switch to r_3, instead of going to r_2 first and then take the long road to n_3.

One might think that this can only happen if there is only one point close to reservoir 2. However the example below shows that this is not true.

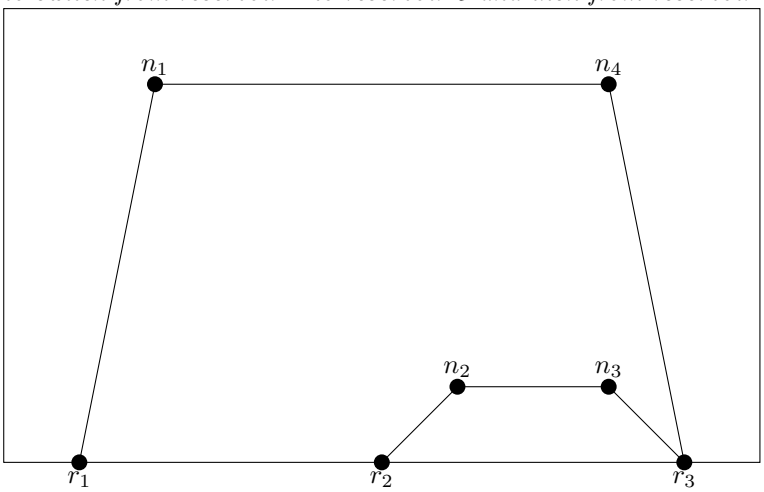
Example 7.



Once again, it is cheaper to use n_3 and n_4 as a switch between r_1 and r_2 instead of stopping by at r_2 and taking the long road to n_5

Before proving any lemmas another example must be given. In the previous examples, all the switches went from one reservoir to another one directly next to it. One might expect that this will always be the case, because switching to a reservoir further away and then come back later to the reservoir in between sounds sub-optimal. The next example shows that in some cases, this still is the optimal choice.

Example 8. *The next example has 4 customers and it is easy to check that it is indeed optimal to switch from reservoir 1 to reservoir 3 and then from reservoir 3 to reservoir 1.*



The above examples give a lot of information about the requirements of an algorithm. A first idea would be to construct an algorithm the same way as before, but this will require some alterations. First of all, an algorithm must check every possible combination of switches between reservoirs. It does not suffice to just check switches from reservoir 1 to 2 and from 2 to 3, because in that case, the algorithm will never find the above solution. Secondly the algorithm needs a way to check if a reservoir is going to be used or not. Simply connecting customers to their closest reservoir will not suffice anymore. It is therefore wise to find a requirement that indicates when a reservoir will be used for sure. This will be partly done in the next lemma.

Lemma 10. *For all i, if $|N_{r_i}| > 2$, then there exists an optimal solution that uses r_i.*

Proof. Assume that $|N_{r_i}| > 2$ and assume that there is an optimal solution that does not use r_i. It is known that there are at least three points close to r_i, call these n_1, n_2, n_3. There are now a few possibilities. Either there is no switch, there is one switch, or there is more than one switch.

If there is no switch, then only one reservoir r_j is being used, where evidently $i \neq j$. Replacing the edge from any customer n_1, n_2, n_3 to r_j with an edge to r_i will create a switch between r_i and r_j. It is possible to then create a new feasible solution, which will not be more expensive and therefore optimal.

If there is one switch, then at least one of the customers n_1, n_2, n_3 is not being used in this switch. Assume n_1 and assume it is connected to r_j. Replacing the edge between n_i and r_j by an edge to r_i will create a switch from r_i to r_j. It is then possible to create a new feasible solution, because there is already a switch between the other two reservoirs. This solution is then again not more expensive and therefore optimal.

In the last case, it is possible that n_1, n_2, n_3 are all part of a switch, but connecting one to r_i will just create a new switch, while the other switch is still intact. It is then again possible to create a new feasible solution that will be optimal again.

This proves the lemma. □

This lemma proves that when there are at least three points close to a reservoir, there exists an optimal solution that uses that reservoir. However, it does not prove anything about the usage of the other reservoirs in the same solution. The next lemma will prove something that will help with this.

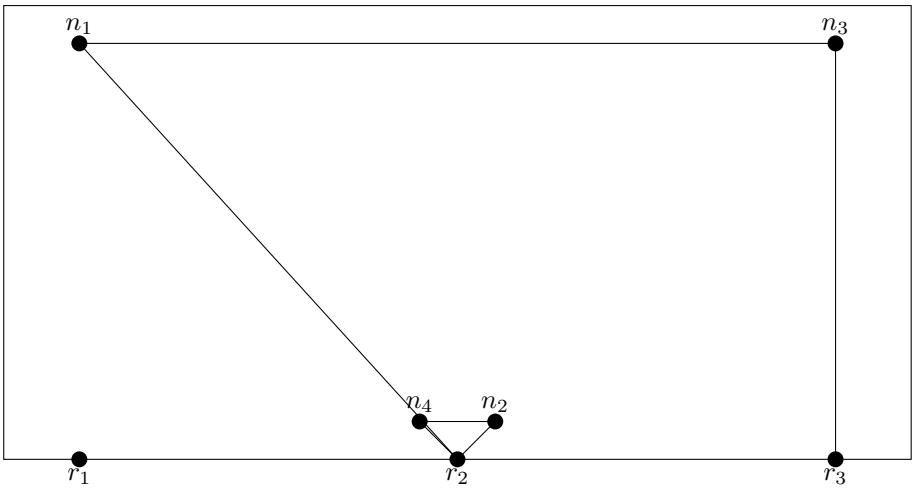
Lemma 11. *If for all i, $|N_{r_i}| > 2$, then there exists an optimal solution that uses all reservoirs.*

Proof. Assume that for all i, $|N_{r_i}| > 2$ and that there exists an optimal solution that does not use all reservoirs. Then either one or two reservoirs are being used in the optimal solution.
Assume first that there are two reservoirs being used. This means that there is a switch between these reservoirs. Assume that r_i is not being used. There are three points close to r_i. Then either all points are contained in a switch, which means there is more than one, or there is at least one point that is not contained in a switch. In the first case, replacing one of the edges will create a switch from r_i to one of the other reservoirs, while still leaving one switch between the other reservoirs. This means that the new solution uses all three reservoirs and it is easy to check that it is feasible and not more expensive and therefore optimal. In the second case, one of the points not used in a switch should be connected to r_i. This will create a switch to one of the other reservoirs, while still preserving a switch between the other two. This will again lead to an optimal solution that uses all three reservoirs.

Now assume that there is only one reservoir being used. Without loss of generality, assume r_1 is not being used. If it is being used, look at r_3, the other side reservoir. There are three points close to r_1. It is possible to create a feasible solution after connecting one of these customers to r_1. This creates a switch between the two reservoirs and after some altering will result in a feasible solution. This solution is not more expensive and must therefore be optimal again. In this case, the optimal solution uses two reservoirs again and it was already proven that this can be altered to an optimal solution that uses all three reservoirs. □

One might be inclined to think that when there is at least one point close to one of the side reservoirs that is not in the neighborhood of any other reservoir, that this side reservoir has to be used. Intuitively, one could think that using a simple interchange argument can prove this. However, this is not true and the following example will show this.

Example 9.



The example has again four customers. The coordinates of the customers are $n_1 = (0, 11), n_2 = (9, 1), n_3 = (11, 1), n_4 = (20, 11)$. *The coordinates of the reservoirs are* $(0, 0), (10, 0), (20, 0)$. *It requires a lot of calculations, but the solution shown is truly the optimal solution. However, one can not interchange tours to connect* n_1 *to* r_1 *or use* r_1 *in any other way. Even worse, there does not exist an optimal solution that uses both* r_1 *and* r_3.

This gives an indication on when all reservoirs will be used, but it still doesn't give a definite conclusion on which reservoirs are being used in specific situations. The following exceptions should be considered:

- For all i, $|N_{r_i}| < 3$;
- There exists i such that $|N_{r_i}| < 3$ and for all $j \neq i$, $|N_{r_j}| > 2$;
- There exists i, j such that $|N_{r_i}| < 3$, $|N_{r_j}| < 3$ and for all $k \neq i, j$, $|N_{r_k}| > 2$.

So either one, two or three reservoirs do not have enough points close to them and the others have. In these cases, it is not clear which reservoirs should be used. Since lemma 9 was proven, a lot of these special cases have already been analyzed. For the remaining cases it is not yet clear what an optimal solution would look like and which reservoirs have to be used and if one can even describe the optimal solution in some way. It can at least be assumed that both side reservoirs have at least one customers in their neighborhood that is not close to any other reservoir. Lemma 9 assures that this is a safe assumption.

One should notice, that every exception will use at least two reservoirs. This is because the side reservoirs both have one customer close to them. This means that already these two reservoirs should be used in this minimal case. If one adds more customer, some edges might have to be interchanged, like in example 9, but there will always be at least two reservoirs that are being used. That means that for every exception, one can first try to force one switch and then try to force two switches and then solve the problem the same way as in previous sections. In the more massive problems with more than three customers close to every reservoir, the algorithm will need to force two switches too. Forcing one switch costs slightly less time and the time needed for solving the exceptions will therefore be dominated by the time needed to solve the big problems. The running time of the algorithm will therefore still be the worst case scenario of the big problems, but one now knows that the small exceptions can also be solved in polynomial time.

One could argue that the exceptions are small and that the running time is therefore somewhat insignificant, because the bigger problems will always cost way more time. This is not necessary true, because one should notice that for the second and third exception, the problems could become as large as one wishes it to be. The argument that the running time is insignificant, because the problems are small is therefore incorrect.

Just like previous sections, single tours can not be ignored. With three reservoirs, the possibilities for single tours increase a lot. Once again, two single tours can not be connected to the same reservoir, because these could always be replaced pairwise. Furthermore there is the possibility of a single tour switch. If there is an even number of customers, this gives the option of two single tours each connected to a different reservoir or a single tour switch and a single tour connected to the reservoir that is left open. If there is an odd number of customers, then there are either three single tours connected to a different reservoir, one single tour connected and there is the possibility of a single tour switch. These are all the possible scenarios and the algorithm has to account for them all. It is now possible to construct the algorithm which will be done in the next section.

3.3.2 Constructing the algorithm

Thanks to all the work done above it is now possible to construct an algorithm. The algorithm will first check what reservoirs will be used for sure and if some reservoirs will not be used at all.

The algorithm that was written down here is very limited. This is because there is an enormous amount of exceptions. Writing down all exceptions and what to do in each different case would require an enormous amount of text and would not be very insightful, nor would the algorithm become any clearer. If one would want to know what to do in specific situations, then one can find all the needed information in the previous sections.

Furthermore the pseudocode is not really elaborate. The steps taken when $|N|$ is even or odd are more extensive than described. One should take notice of this when trying to write a code out of this pseudocode.

Input: A set N of customers, locations for r_1, r_2 and r_3
Initialize: An empty ordered set P

Compute N_{r_1}, N_{r_2} and N_{r_3}
Check for all i if $N_{r_i} = N$
Check if $N_{r_1} \cup N_{r_2} = N$ or $N_{r_2} \cup N_{r_3} = N$
Check all special cases concerning the size of each N_{r_i}

if Any of the above holds **then**
    Use one of the algorithms given in previous sections
else ▷ From here one can assume that all reservoirs will be used

    Check if $|N|$ is even or odd

    if $|N|$ is even **then**
        Force the appropriate amount of switches and single tours
        Use Blossom V to match all remaining customers
    else
        Force the appropriate amount of switches and single tours
        Use Blossom V to match all remaining customers
    end if
end if

An ordered set P, which will be an optimal path

3.3.3 Proving optimality

Proving that this algorithm is optimal is mostly done in the previous sections. In these sections, the possible optimal solutions were analyzed and it was concluded that an optimal solution will always be of some form in different situations. The algorithm will first account for all special situations in which it is possible that some reservoirs will not be used. If none of these situations occur the algorithm will proceed to construct an optimal solution that uses all reservoirs. It will force an appropriate amount of single tours and switches and try every possible optimal solution. This means that eventually the algorithm will come across the optimal solution and therefore the algorithm will return an optimal solution.

The running time is a bit more complicated because of the special cases. However, it was already assumed that the running time of the special cases was dominated by the running time of the regular problems. In this case one can use the same logic that was used in previous sections. The running time is then dominated by the case that there is an odd number of customers. The algorithm needs to force two switches, for which there are $O(|N|^4)$ possible pairs of switches. Then it is possible to have three single tours, one single tour or one single tour switch. The running time is dominated by the first one. There are $O(|N|^3)$ possible combinations of single tours. Then the algorithm needs to find an optimal matching for the remaining customers, which will take $O(|N|^5)$ like usual. The worst case scenario will therefore have a running time of $O(|N|^{12})$, which is still polynomial and therefore the problem lies in P.

3.4 k reservoirs, two components

The last section showed that more reservoirs can lead to a lot of exceptions and difficulties. This section will be dedicated to proposing a way to find an algorithm that can solve these problems to optimality for any fixed k. First of all, it is possible to determine the amount of single tours and single tour switches that an optimal solution can have. This is easy and the following scenario can occur for a fixed k.

- If $k, |N|$ are both even, then an optimal solution can have $k - 2l$ single-tours, where $l = 0, 1, \ldots, \frac{k}{2}$. Also if a solution has $k - 2l$ single tours and $l > 0$, then it can have $l - 2m$ single-tour switches where $m = 0, 1, \ldots, \frac{l}{2}$ if l is even and $l + 1 - 2m$ single tour switches where $m = 1, \ldots, \frac{l+1}{2}$ if l is odd;
- If k and $|N|$ are both odd, then an optimal solution can have $k - 2l$ single tours, where $l = 0, 1, \ldots, \frac{k-1}{2}$. Also if a solution has $k - 2l$ single tours and $l > 0$, then it can have $l - 2m$ single tour switches where $m = 0, 1, \ldots, \frac{l}{2}$ if l is even and $l + 1 - 2m$ single-tour switches where $m = 1, \ldots, \frac{l+1}{2}$ if l is odd;
- If k is even and $|N|$ is odd, then an optimal solution can have $k-2l+1$ single tours, where $l = 1 \ldots, \frac{k}{2}$. Also if a solution has $k - 2l$ single tours and $l > 0$, then it can have $l - 2m$ single tour switches where $m = 0, 1, \ldots, \frac{l}{2}$ if l is even and $l + 1 - 2m$ single-tour switches where $m = 1, \ldots, \frac{l+1}{2}$ if l is odd;
- If k is odd and $|N|$ is even, then an optimal solution can have k, where $l =$ .Also if a solution has $k - 2l$ single tours and $l > 0$, then it can have $l - 2m$ single tour switches where $m = 0, 1, \ldots, \frac{l}{2}$ if l is even and $l+1-2m$ single tour switches where $m = 1, \ldots, \frac{l+1}{2}$ if l is odd.

This gives an indication on what an optimal solution could look like. An algorithm that will try to find an optimal solution will have to account for any of these scenarios.

The last section was mostly dedicated to determining which reservoirs are to be used and which ones do not have to be used in an optimal solution. One should check that lemma 10 and 11 still hold with the new amount of reservoirs. Even if lemma 11 holds, the number of exceptions that still need to be checked increase too. One should check if they can be solved in reasonable time or not. More reservoirs means a lot more exceptions and solving them might become immensely difficult and complex.

However, one can still assume that at least two reservoirs are being used in the special cases. Therefore an algorithm could try to force one switch at first and increase this number while it searches for the optimal solution. The optimal solution must use at least one switch, so at one point the optimal solution is found. Due to the nature of the running time, checking if the algorithm needs k switches still costs the most time and it dominates the other running times. There are $O(k!|N|^k)$ different switches if k switches are assumed. Then an algorithm needs to check all possible single tours and find an optimal matching. The running time of these algorithms would therefore be at least $O(k!|N|^k)$. For any fixed k, this can still be polynomial. However, if k is considered part of the input, then one sees that this becomes exponential in k. One can therefore conclude that this algorithm is not good enough to proof that this problem is in P. Another algorithm might prove this, but one can not be sure about that with the information given in this thesis.

3.5 One reservoir, k components

Some other variation is when the amount of components is increased to a fixed amount k. This is a special case of the k-tour cover problem. In this problem, there is also a set of customers and there is an origin. The goal of the problem is to find a set of k-tours, which is a tour of at most k points, that is as cheap as possible and covers all customers. The difference is that the location of the so called origin does not have any requirements, where it does have some limitations in the current setup.

The k-tour cover problem is a problem that is NP-hard if $k > 2$. The problem is even APX-complete for $2 \leq k \leq |N|$, which means that there is not even a PTAS that solves the k-tour cover problem. For larger k the problem can be reduced to TSP for which there do exist a PTAS. When this problem is put into a two dimensional Euclidean space there do exist a PTAS. The article written by Tetsuo Asano [6] proves that the k-tours covering problem is APX complete. The k-tours covering problem is a variation of the Capacitated Vehicle Routing Problem. An added constraint is that the demand of customers can be split in integral parts. The problems that are described in this thesis are a specific subset of these set of problems.

Furthermore, an article has been written by M. Khackay[8], in which he analyzes a variant of this problem in a Euclidean 3-dimensional space. The approximation that is proposed can also be extended to higher dimensions.

The specific problem that this thesis is interested in has a few limitations that do not occur in the Vehicle Routing Problem, but these do not matter when looking at solving this problem in polynomial time. This article does not take a look at the problem with more reservoirs, but looking at the complexity that occurred in section 2 and 3, one should expect that this would not make the problem easier to solve. One can conclude that $\overline{P_R^k}$ is at least NP-hard if $k > 2$ for any R and might be even more difficult in some cases.

4 Free reservoirs, L_1 metric

This chapter will take a closer look at D_C^R for some values of C and R. The main focus of this chapter will first be the location of the reservoirs. In this specific problem the locations have to be chosen and this gives a new layer of complexity to the problem. Although, thanks to the L_1 metric, this will not be that hard.

When this is done, the problem will look very similar to the problems from the previous chapter and the reader may notice a few similarities in the lemmas and the algorithm.

The chapter will give a few polynomial algorithms that solve D_C^R to optimality and will also give the running time of these algorithms. The starting point will again be $C = 2$ and $R = 1$, because this is again the first non-trivial case. After that, R will first be incremented. C will not be incremented, because the problem is very similar to the problem of the last chapter. In this chapter it was seen that increasing the amount of components made the problem at least NP-hard and the same can be expected from this problem.

4.1 One reservoir, two components

This problem will work with a L_1 metric, which will mean that the graph resulting from the distances between customers still has triangle inequality. The first difference is that the location of the reservoirs is not fixed so this must be dealt with first. After that, a few lemmas will be proven and finally an algorithm will be constructed. One will find out that the algorithm runs in polynomial time and therefore the problem is in P. This will also be proven.

4.1.1 Choosing the location of the reservoir

First, one should notice that, thanks to triangle inequality, there can still only be one single tour connected to the reservoir. This means that all other customers will be matched together and that every customer will be connected to the reservoir only once, except for a possible single tour, which will be connected to the reservoir twice.

To optimize the location of the depot, one must optimize the distance functions from the customers to the depot. If there is an odd number of customers one could add a dummy customer at the location of the dedicated single tour. Later on there will be a lemma that this in fact does not matter for the location of the depot.

For now, one should notice that the distance function that has to be optimized is $\sum_{i=1}^{|N|} |x-x_i|+y_i$, where customer n_i has coordinates (x_i, y_i). Expanding this will give $\sum_{i=1}^{|N|} |x - x_i| + \sum_{i=1}^{|N|} y_i$, where the second part is not dependent on the location of the reservoir. This means that the second part is a constant. Therefore the y values of the coordinates of the customers do not matter when optimizing the location of the depot. One should ignore them and concentrate on the x coordinates. With this remark it is now possible to prove the next lemmas.

Lemma 12. *If $|N|$ is even and the customers are numbered from left to right $n_1, n_2, \ldots, n_{|N|}$ in order of x coordinates and $n_i = (x_i, y_i)$, then $x_{\frac{|N|}{2}}$ is the optimal location for the reservoir.*

Proof. This will be proven by induction on k where $|N| = 2k$. If there are zero customers there is nothing that has to be proven, which is why $k = 0$ makes no sense. Assume $k = 1$, which means that there are two customers n_1, n_2, with x coordinates x_1, x_2. It was assumed that $x_1 \leq x_2$

which implies that $s = x_2 - x_1$ will never be negative. There are now three options for choosing the location of the reservoir x_r.

- $x_r < x_1$;
- $x_1 \leq x_r \leq x_2$;
- $x_2 < x_r$.

In all cases look at the sum $|x_r - x_1| + |x_r - x_2|$. The first case gives the following equation:

$$\begin{aligned} |x_r - x_1| &= x_1 - x_r \\ |x_r - x_2| &= x_2 - x_r \\ &= x_2 - x_1 + x_1 - x_r \\ &= s + x_1 - x_r. \end{aligned}$$

Which means that the total distance $|x_r - x_1| + |x_r - x_2| = s + 2(x_1 - x_r)$. Since $x_r < x_1$ this distance is strictly larger than s.

In the last case something similar can be done:

$$\begin{aligned} |x_r - x_2| &= x_r - x_2 \\ |x_r - x_1| &= x_r - x_1 \\ &= x_r - x_2 + x_2 - x_1 \\ &= x_r - x_2 + s. \end{aligned}$$

In this case the total distance becomes $|x_r - x_1| + |x_r - x_2| = s + 2(x_r - x_2)$, which is also strictly larger than s.

In the second case one should notice that $|x_r - x_1| = x_r - x_1$, since x_r is always larger or equal to x_1 and that $|x_r - x_2| = x_2 - x_r$ since x_r is always smaller or equal to x_2. This gives a total distance of $|x_r - x_1| + |x_r - x_2| = x_r - x_1 + x_2 - x_r = x_2 - x_1 = s$. This implies that any location between x_1 and x_2 suffices, since x_1 is one of these locations, the lemma holds for $k = 1$.

Assume that the lemma is proven for $k \geq 1$ then look at the case $k + 1$. First one should note that for the outer customers x_1 and $x_{2(k+2)}$ it would be optimal if the reservoir would be located somewhere in between them. Look at the remaining points $x_2, \ldots, x_{2(k+1)-1}$ and notice that this are exactly k points, for which the lemma was already proven. The optimal location will therefore lie at x_{k+1}, which is in between x_1 and $x_{2(k+1)}$. This means that the location must be optimal when considering all of the points, which means that the lemma holds for $k + 1$.

By induction the lemma must now hold for general $k > 0$. □

This lemma can be slightly improved. In the first part of the proof it is noticed that any location between x_1 and x_2 might be an optimal location for the reservoir. It is still true that for $2k$

customers, any location between x_k and x_{k+1} is optimal. In particular, the remainder of this thesis will use that exactly these points x_k and x_{k+1} are optimal.

The next lemma will prove something similar in case that there is an odd number of customers.

Lemma 13. *When $|N|$ is odd and the customers are numbered $n_1, n_2, \ldots, n_{|N|}$ in order of x coordinates, if $n_i = (x_i, y_i)$, then $x_{\frac{|N|+1}{2}}$ is the optimal location for the reservoir.*

Proof. This lemma will also be proven by induction on k where $|N| = 2k + 1$. Assume that $k = 0$ and there is only one customer n_1. Then it is easy to see that the optimal location of the reservoir is directly beneath n_1. This proves the lemma for $k = 0$.

The induction step of this lemma is the same as used in the previous lemma and will not be done again. □

In both lemmas the customers where numbered in order of x coordinates. It was never explicitly mentioned what has to be done when two customers have the same x coordinates. The proof is not dependent on this and it therefore does not matter how those customers are ordered. If this situation occurs, the customers will be ordered at random.

One more corollary follows from these lemmas, which will be very important when constructing the algorithm at the end.

Corollary 3. *If $|N|$ is odd and an extra customer is added, then the optimal location for the reservoir does not change.*

Proof. Assume that there are $2k + 1$ customers, where k is an non-negative integer. According to the lemma, the optimal location for the reservoir will then be x_{k+1}. Add a customer m at a random location with coordinates (x_m, y_m). Then the following options are possible:

- $x_m < x_{k+1}$;
- $x_m > x_{k+1}$;
- $x_m = x_{k+1}$.

In all cases, the customers have to be numbered again and it will turn out that the optimal location of the reservoir does not change.

In case one, the customers can be numbered in a way such that the new index of customer x_{k+1} will become $k + 2 = \frac{2k+2}{2} + 1$, which was proven to be an optimal location.

In case two, the customers can be numbered such that the new index becomes $k + 1 = \frac{2k+2}{2}$, which was also an optimal location.

In the last cases, the customers can be numbered in different ways, but one could number the customers in a way such that the new index becomes $k + 1$, which was already proven to be optimal before. This proves the lemma. □

This completes all things that must be said about the location for the reservoir for now. The tools provided in earlier sections also come in handy at this moment. The same lemmas that were true in section 3.1 are also true in this section. This is because most of those lemmas only

depend on triangle inequality, which is still present in the current problem. The proofs might have to be altered slightly, but to do so would not be very insightful and will therefore be left to the reader.

Picking the location is the only aspect that really changed in this problem, but with the information that was gathered above, an algorithm can be constructed without any further work.

4.1.2 Constructing the algorithm

The algorithm will look very familiar, since it will use the same approach as was used in section 3.1. The only difference is that the algorithm will try to find the best possible location for the reservoir first. From the lemmas above, it is known for both an even and an odd number of customers what the optimal location will be.

In the case of an odd number of customers, the algorithm will try to find the cheapest solution with one single tour. One might think that this would alter the location of the reservoir, but as proven above, this is not the case. This makes the algorithm really straightforward.

Input: A set N of customers

Initialize: An empty ordered set P

Number all customers in order of x coordinates

if $|N|$ even **then**
  Set $r = (x_{\frac{|N|}{2}}, 0)$
  Blossom V: Find optimal matching M of N
  Number edges $e_1, \ldots, e_{\frac{|N|}{2}}$ in M
  for $i = 1, \ldots, \frac{|N|}{2}$ **do**
    Add $(r, v_i), (v_i, w_i), (w_i, r)$ to P, where $e_i = (v_i, w_i)$
  end for

else
  Set $r = (x_{\frac{|N|+1}{2}}, 0)$
  for all $n \in N$ **do**
    Blossom V: Find optimal matching M of $N - \{n\}$
    for $i = 1, \ldots, \frac{|N|}{2}$ **do**
      Add $(r, v_i), (v_i, w_i), (w_i, r)$ to P, where $e_i = (v_i, w_i)$
    end for
    Add $(r, n), (n, r)$ at the end of P
    Check if this solution is cheaper then the previous saved one, save it if it is
  end for

end if

Output: An ordered set P, which will be an optimal path

4.1.3 Proving optimality

The proof that the algorithm is optimal is really straightforward, since it is very similar to the proof of section 3.1. It might be unclear if it isn't possible to move the reservoirs in a way that makes the solution cheaper. However, the optimal location was proven and the algorithm starts by setting the reservoir at this location. The algorithm will then find the optimal matching and return this when there is an even number of customers. This is optimal, since there is no place where one could optimize anything. If there is an odd number of customers the algorithm will try every possible single tour and find the optimal solution for that single tour. At one point it must come across the optimal solution. This proves that the algorithm is optimal and will at one point find the optimal solution.

The running time of this algorithm is the same as the running time of the algorithm in section 3.1, namely $O(|N|^6)$. This happens, because the only real difference is that this algorithm numbers all customers and puts the reservoir in place. There is nothing else that really changed in the algorithm. Putting the reservoir in place is no work at all and sorting all the customers can be done in less time. This running time is polynomial and therefore this specific problem is part of P.

4.2 Two reservoirs, two components

The next section will expand on the first by again adding a reservoir, just like in the previous chapter. This still looks very similar to chapter 3 but the algorithm will be modified a bit. This is due to the fact that the location for both reservoirs have to be chosen. The first subsection will be dedicated to making some remarks about this problem that will come in handy later. Then another algorithm will be constructed and it will be proven that this problem can also be solved to optimality in polynomial time.

4.2.1 Preliminary results

The general idea of the algorithm that will be constructed at the end of the chapter is to take a subset of customers and connect them to the first reservoir and then connect the others to the other reservoir. Then a switch will be chosen just like before and all other customers will be matched optimally. When all customers from N are ordered by their x coordinates, it will be shown that there are only a few possible subsets that need to be considered. The subsets that need consideration are of the form $\{n_1, \ldots, n_k\}, \{n_{k+1}, \ldots, n_{|N|}\}$ for different values of k. It will be shown later that these are indeed the only subsets that need consideration.
The next lemma will first exclude already two of these possible subsets.

Lemma 14. *There exists an optimal solution that uses all reservoirs.*

Proof. Assume that there is only one customer, then reservoir 1 and 2 can both be located directly beneath it. In this case the solution $(r_1, n_1), (n_1, r_2)$ is then a feasible solution and it is also optimal. It must be noted that only r_1 could have been used and r_2 could have been ignored, but this does not matter for the lemma.

Assume that there are $k > 1$ customers and assume that there is an optimal solution that uses only one reservoir r_1. This means that there is a path that traverses all customers and eventually ends in r_1. Take a look at the last edge traversed and assume that this edge is (n, r_1). If r_2 is now located directly beneath n and the last edge is replaced by (n, r_2), then this still gives a feasible solution. Since r_2 is directly beneath n, this solution will also surely not be more expensive than the previous one. Therefore it must be optimal. This completes the proof and thus the lemma holds. □

As mentioned before, this lemma makes sure that the pair of subsets $(N, \emptyset), (\emptyset, N)$ don't have to be considered when looking for an optimal solution, because there always exists an optimal solution that uses both reservoirs.

It has also been mentioned before that there is only a specific amount of subsets that need to be checked. One should make sure that there is no possibility that the optimal solution uses another pair of subsets that is not considered. This will be proven next, but to do so the following lemma is needed first.

Lemma 15. *When all customers are ordered based on x coordinates, then for all $k < l$ either $|x_{r_1} - x_k| \leq |x_{r_2} - x_k|$ or $|x_{r_2} - x_l| \leq |x_{r_1} - x_l|$.*

Proof. Assume that $k < l$ and assume that the lemma does not hold. Then $|x_{r_1} - x_k| > |x_{r_2} - x_k|$ and $|x_{r_2} - x_l| > |x_{r_1} - x_l|$. Also it is clear that $x_{r_1} \leq x_k \leq x_l \leq x_{r_2}$, because if x_l, x_k would not lie inside this interval, so it would be clear that the lemma would hold. Then the following is

true:

$$\begin{aligned} |x_{r_2} - x_{r_1}| &= |x_{r_2} - x_l| + |x_l - x_k| + |x_k - x_{r_1}| \\ &> |x_l - x_{r_1}| + |x_l - x_k| + |x_{r_2} - x_k| \\ &> |x_{r_2} - x_{r_1}|. \end{aligned}$$

This is clearly a contradiction and therefore the lemma holds. □

This lemma is needed to save some space in the next proof. The next lemma will prove that indeed it suffices to look at specific subsets and ignore all other ones.

Lemma 16. *If all customers are ordered by x coordinates, then there exists an optimal solution in which there exists a $k \in \mathbb{N}$, such that $n_1, \ldots, n_k$ are matched to reservoir r_1 and $n_{k+1}, \ldots, n_{|N|}$ are matched to reservoir r_2.*

Proof. The proof will show that it is always possible to construct such an optimal solution out of an other optimal solution that is not of this form.

Assume that there is an optimal solution in which the properties do not hold. Then there exist k, l with $k < l$ such that n_k is matched to r_2 and n_l is matched to r_1. If there exists such a k, there must also be a smallest one and it is assumed that k is from now on the smallest k possible. The same will hold for l and it is assumed that l is the smallest one possible.

Assume that $n_i = (x_i, y_i)$. From the previous lemma, it is known that $|x_{r_1} - x_k| \leq |x_{r_2} - x_k|$ or $|x_{r_2} - x_l| \leq |x_{r_1} - x_l|$. Assume first that $|x_1 - x_k| \leq |x_2 - x_k|$ holds.

There are now a few possible scenario.

- Both $(n_k, r_2), (r_2, n_k)$ are elements of P^*;
- There exists an m such that $(r_2, n_k), (n_k, n_m)(n_m, r_2)$ or $(r_2, n_m), (n_m, n_k), (n_k, r_2)$ are elements of P^*;
- There exists an m such that $(r_2, n_k), (n_k, n_m)(n_m, r_1)$ or $(r_1, n_m), (n_m, n_k), (n_k, r_2)$ are elements of P^* and n_m is right from n_k;
- There exists an m such that $(r_2, n_k), (n_k, n_m)(n_m, r_1)$ or $(r_1, n_m), (n_m, n_k), (n_k, r_2)$ are elements of P^* and n_m is left from n_k;
- Either $(r_2, n_k), (n_k, r_1)$ are elements of P^* or $(r_1, n_k), (n_k, r_2)$ are elements of P^*.

It is easy to check that these are the only possibilities, since n_k has to be somehow attached to r_2. It can only be a single tour or a regular tour, which limits the possibilities to the ones above.

In all above cases, a new optimal solution will be constructed in which n_k is not connected to r_2 anymore. This will mean that either there is another $k < l$ such that the same situation occurs, or all $k < l$ are dealt with and one could search for new l for which the situation occurs. The idea is to keep constructing new optimal solutions till the point that the optimal solution is of the desired form.

In the first case n_k is just a single tour attached to r_2. It is not a switch and it is already known that r_1 is also being used, since n_l is attached to it. Replacing both edges by edges connected

to r_1 will still give a feasible solution and since $|x_1 - x_k| \leq |x_2 - x_k|$ was assumed, this solution must be optimal again. The new path has to be altered slightly, but it should be very intuitive to do this.

In the second case m must be larger than k, because it was assumed that k was the smallest possible value. Also, this tour is not a switch. Replacing the edge between r_2 and n_k will create a switch between r_1 and r_2. It is already known that there is a switch, since r_1 and r_2 are both being used. This new switch can be traversed at the end of the optimal path that was already used in the old solution. The direction of this switch must be chosen then, but this should not give any problems.

In the third case there is a switch, but it easy to see that switching the role of m and k in this switch gives again a feasible solution that is surely not more expensive then the original one and therefore must again be optimal.

In the last two cases n_k is contained in a switch that is not easily removed without creating reasonable doubt about the feasibility of the new solution. In both cases it might be possible that there exists at least one other switch. If that is the case, then n_k can be connected to r_1. This would require some alterations of the path, but since there exists a switch, it is possible to again create a feasible solution. This solution will not be more expensive then the previous one and will therefore be optimal.

Assume that there does not exist another switch. Then there are two possible scenarios:

- $(n_l, r_1), (r_1, n_l)$ are elements of P^*;
- there exists an s such that $(r_1, n_l), (n_l, n_s)(n_s, r_1)$ or $(r_1, n_s), (n_s, n_l), (n_l, r_1)$ are elements of O^*.

These are the only scenarios, because n_l can not be part of the same tour as n_k, since m was assumed to be smaller than l. Therefore n_l can not be part of a switch, but it must be connected to r_1. This only leaves the options above.

Assume that n_l is part of a single tour and assume that n_k is not part of a single tour switch. Without loss of generality assume that (r_2, n_k) is being used. Replace this edge with (r_1, n_k) and replace (r_1, n_l) by (r_2, n_l). One should notice that this gives a new feasible solution in which there is again one switch. This solution is not more expensive then the previous one and therefore it must be optimal again. The case where (n_k, r_2) is being used is similar. Now assume that n_k is part of a single tour switch. A previous lemma has proven that this can not be the case, since then there would be two single tours attached to r_1, which is a contradiction.

Assume that n_l is not part of a single tour and assume that n_k is part of a single tour. One should note that in either cases $s > l$ or $s < k$, because it can not lie in between k and l, because this would be a contradiction with the assumption that k and l are minimal. Assume that $s > l$. Then it is cheaper to connect n_s to r_2 and to make a single tour out of n_k connected to r_1 instead of a switch. It is possible that the direction of this switch has to be altered, but this will still give a feasible solution that is also optimal. If however $s < k$, then it is possible to do the same by interchanging n_l and n_s in the above argument.

Assume now that n_k is not part of a single tour. Still it must be true that either $s > l$ or $s < k$. Assume that $s > l$. Replace now (r_2, n_k) by (r_1, n_k) and replace the tour from (r_1, n_s) by (r_2, n_s). Since n_k lies left from n_s, it will never be more expensive to interchange these edges, because the replacement of the edge from n_k is never more expensive then the original edge from n_s. The same can be said about the other replacement. This new solution still has a switch and it is therefore possible to construct a feasible solution out of these edges, which will then again be optimal. Now assume that $s < k$. The same procedure can be used except using n_l instead of n_s. This will result in the same situation as above and therefore this will also give a new optimal solution.

This completes the part where was assumed that $|x_{r_1} - x_k| \leq |x_{r_2} - x_k|$. Under this assumption it is now always possible to connect n_k to r_1, instead of to r_2. Next assume that $|x_{r_2} - x_l| \leq |x_{r_1} - x_l|$. This will give the same five scenarios as above. The proof of the last three scenarios did not actually use the assumption that $|x_{r_1} - x_k| \leq |x_{r_2} - x_k|$, but used that n_l was right from x_k. These proofs would not change when the assumption is altered and shall therefore be skipped.

The first two scenarios did use the assumption and a new way to construct a new optimal solution should be figured out. Assume that n_k is part of a single tour connected to r_2. Simply connecting it to r_1 will not suffice, since this might increase the cost of the solution and therefore damage the optimality. Instead look at n_l. There are a few possible scenarios.

- Both $(n_l, r_1), (r_1, n_l)$ are elements of P^*;
- There exists an s such that $(r_1, n_l), (n_l, n_s)(n_s, r_1)$ or $(r_1, n_s), (n_s, n_l), (n_l, r_1)$ are elements of P^*;
- There exists an s such that $(r_1, n_l), (n_l, n_s)(n_s, r_2)$ or $(r_2, n_s), (n_s, n_l), (n_l, r_1)$ are elements of P^* and n_s is right from n_l;
- There exists an s such that $(r_1, n_l), (n_l, n_s)(n_s, r_2)$ or $(r_2, n_s), (n_s, n_l), (n_l, r_1)$ are elements of P^* and n_s is left from n_l;
- Either $(r_1, n_l), (n_l, r_2)$ or $(r_2, n_l), (n_l, r_1)$ are elements of P^*.

In all cases it is now possible to connect n_k not to r_1, but to r_2. The proofs of this are really similar and will not be done.

When n_k is part of a tour connected to r_2, the same can be done with n_l, but this will also not be done, because it is very similar to what was already done.

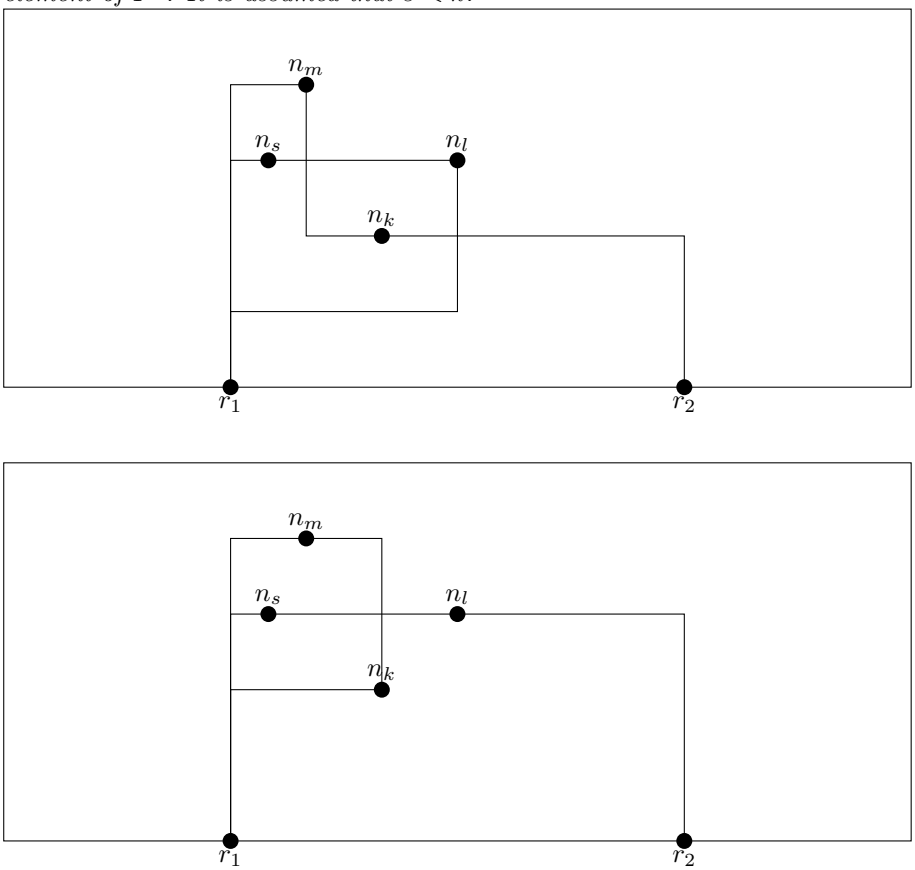
If there exists $k < l$ such that n_k is connected to r_2 and n_l is connected to r_1, it is now always possible to either connect n_k to r_1 or to connect n_l to r_2. When this is done for minimal k and l, it will effectively increase either k or l or both. Since there is only a finite amount of customers, this means that at some point there will be an optimal solution in which the conditions of the lemma holds. This means that the lemma is proven. □

This lemma proves that there always exists an optimal solution in which two reservoirs will not cross each other. This means that if n_k is connected to r_1 and n_l connected to r_2, then $k < l$. One very important remark should be noted, which is the remark that a single tour switch can, and will in some cases, still exist. Also, the proof gives the exact location of where the single tour switch will be located, namely precisely the end point of the first subset, or the first point of the

second subset. In all other cases some customers would be crossing each other, which could be removed as was proven above. Therefore it should be noted that in the case of an odd number of customers, there are more subsets that need to be checked, namely the subsets that overlap exactly one point in the middle. These are the pairs of subsets $\{n_1, \ldots, n_i\}, \{n_i, \ldots, n_{|N|}\}$. The algorithm constructed at the end of this thesis will also check these subsets.

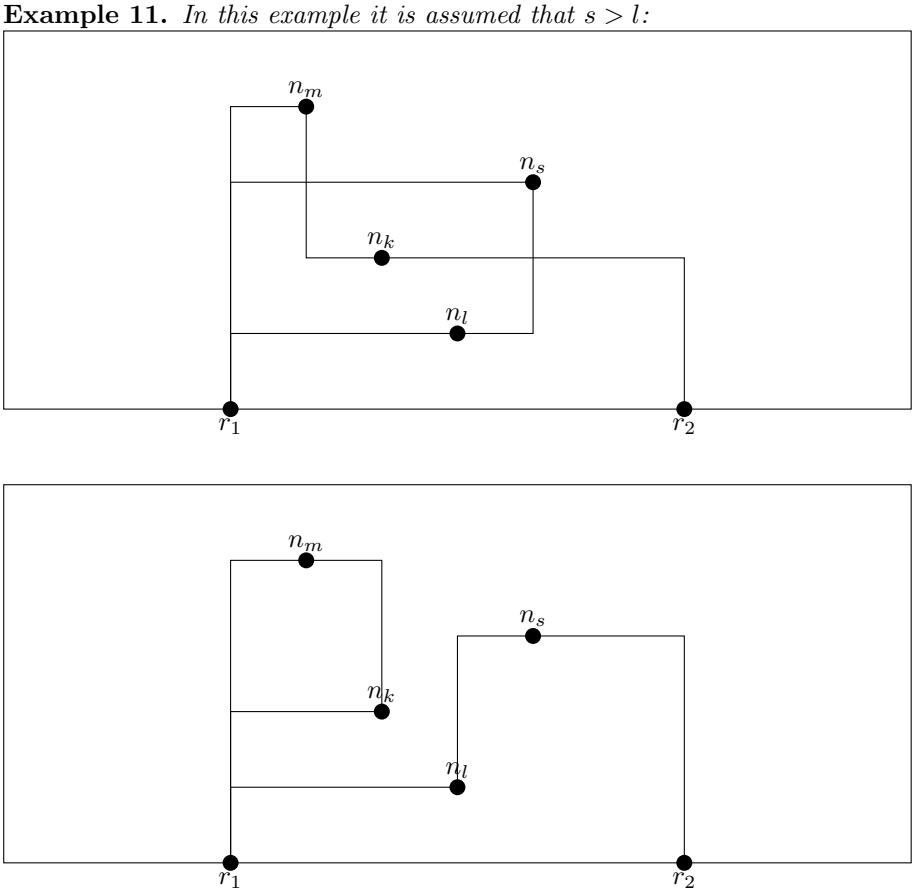
The above proof was very long and it might be wise to look at some examples of situations that were analyzed in the proof. This might give a better intuition about some of the processes that were going on and about the proof itself.

Example 10. *This example is the case where n_l is not part of a single-tour and (r_2, n_k) is an element of P^*. It is assumed that $s < k$:*



The first picture shows the situation where n_l is connected to r_1 and n_k to r_2. In the second situation it is clearly done in a better way, but there still is a switch constructed, which is needed for feasability. This is cheaper, because the solution now effectively skips the distance between n_k and n_l.

Example 11. *In this example it is assumed that $s > l$:*



Lemma 16 is extremely useful. It is now possible to start looking at the subsets that were indicated before, because the lemma guarantees that there exists an optimal solution in which these subsets are used. This means that the algorithm does not have to worry that there only exist optimal solutions in which some customers cross each other, since lemma 16 assures that this will never happen.

Something should be said about single tours. If there was only one reservoir, it was proven that adding an extra point to an odd number of customers would not change the optimal location of the reservoir. This was very beneficial, because adding a customer would simulate a single tour, which was needed if the number of customers was odd. If there are two reservoirs, the algorithm will split the number of customers in two groups and therefore one of them will be even when $|N|$ is even. It is however possible that the single tour exists on the side of the even subset. Therefore the algorithm will try to add a point to that set to simulate this. However this will in some cases change the optimal location of the reservoir. Fortunately, this is not a real issue. In the first section, two optimal locations were given and it is easy to see that adding a point to the set of customers will change the optimal location to either one of these two. It is also easy to see that adding a point to the right of the middle will use the right optimal location choice and adding a point to the left will use the left. This could be proven in a lemma, but this is not really insightful and will therefore be skipped. The case of a possible single tour was already discussed. For this possibility one needs to check the extra subsets of the form $\{n_1, \ldots, n_i\}, \{n_i, \ldots, n_{|N|}\}$.

Checking these extra subsets will actually not cost much extra time, since these subsets are specifically used to simulate the single tour switch, which will be the customer n_i. Therefore the algorithm does not have to check all possible single tours, which will actually save a factor $O(|N|)$. The time needed to calculate the remaining possible solutions is dominating the time to check the single tour switches.

The discussion about single tours and the lemma that was proven are enough to start constructing an algorithm.

4.2.2 Constructing the algorithm

The next algorithm will start looking at the specific subsets. For every possibility it is known that there should be a switch and the algorithm will try to force all switches. The other customers are then matched to optimality. Furthermore the algorithm will also try to force single tours. If $|N|$ is even, it will try to force a single tour for each reservoir. In the case that the number of customers is odd it will try to force a single tour for either reservoir. Also, if the number of customers is odd, it will try every customer to be the single tour switch, which are indicated by two very specific subsets as was stated above.

Input: A set N of customers

Initialize: An empty ordered set P

Order all customers of N according to x coordinates

Check if $|N|$ is odd or even

Set: $N_L = \{n_1\}$,$N_R = N - \{n_1\}, N'_L = \{n_1\}, N'_R = N$, $k = 1$

while $N_L \neq N$ **do**

if $|N|$ is even **then**

for all $n \in N_L, m \in N_r$ **do**

for all $a \in N_L, b \in N_r$ **do**

Locate the optimal locations for r_1, r_2 accounting for single tours a, b

Find the optimal matching of N without these points

Add all resulting tours to P

Check if P is cheaper than the previous solution and save it if it is

Do the same without the accounting for the single tours.

end for

end for

else if $|N|$ is odd **then**
for all $n \in N_L, m \in N_r$ **do**
for all $a \in N$ **do**
Locate the optimal location for r_1, r_2 accounting for single tour a
Find the optimal matching of N without these points
Add all resulting tours to P
Check if P is cheaper than the previous solution and save it if it is
Do the same where a is used as a single tour switch
end for
end for
end if
Add the first point from N_R to N_L
end while

if $|N|$ is odd **then**
while $N'_R \neq \emptyset$ **do**
Locate the optimal location for r_1, r_2
Find the optimal matching of $N - \{n_k\}$
Add all resulting tours to P
Check if P is cheaper than the previous solution and save it if it is
Remove n_k from N'_R, add n_{k+1} to N'_L, increment k by one
end while
end if
Output: An ordered set P, which will be an optimal path

4.2.3 Proving optimality

The last lemma that was proven helps a lot in proving the optimality of the algorithm. It is now a fact that there exists an optimal solution that uses any of the pair of subsets $\{n_1, \ldots, n_i\}, \{n_i, \ldots, n_{|N|}\}$. It is then only left to prove that, if these subsets come around, the algorithm will find the optimal solution. In this situation both reservoirs are being used, which means that there always is a switch. The algorithm will try every possible switch, knowing that such a switch will go from a customer in N_{r_1} to a customer in N_{r_2}. Therefore, the algorithm will at some point come across the switch that should be used in an optimal solution. In this case it is possible that there are single tours, but the algorithm will also account for that. Then all that is left is to find an optimal matching and the algorithm will do this by using Blossom V. This means that at some point the algorithm will come across the cheapest solution using a specific switch and having some single tours or not. If the optimal location uses a single tour switch, the last part of the algorithm will try all possible single tour switches. These are limited because of the same lemma and the algorithm will always find the optimal solution for every single one of the single tour switches. Therefore, if the optimal solution uses a single tour switch, the algorithm will find it. One can conclude that in any case, the algorithm will eventually find an optimal solution.

Something should be said about the running time of this algorithm. The algorithm will always try a few pairs of subsets. There are a total of $O(|N|)$ pairs of subsets. For every subset, the algorithm will try every possible switch, of which there are $O(|N|^2)$. If there is an even number of customers, the algorithm will try two single tours. There are also $O(|N|^2)$ pairs of single tours. If N is odd there is only one single tour, so the algorithm only has to check $O(|N|)$ of them. Then the only thing left is to find an optimal matching, which can be done in $O(|N|^5)$ time. This means that the worst case scenario occurs if $|N|$ is even. It will take $O(|N|^{10})$ time to solve the problem to optimality. One should note that setting the reservoirs and adding all tours to P does not add up much to the time and can be ignored. The running time is therefore $O(|N|^{10})$, which is still polynomial. The problem therefore lies in P.

4.3 Three reservoirs, L_1 metric

In the following section, another reservoir is added. The main idea that will be discussed in the next section is to construct an algorithm in the same way as was done in the previous chapter. This will require an algorithm to look at specific subsets of the customers and try to find the best possible solution for these subsets. For this idea to work, it is crucial that lemma 16 can be extended to hold in case of three reservoirs. This will be the main focus of the setup section. After that, the algorithm will be constructed and it will be proven that this indeed solves the problem to optimality. A running time will also be given and it will turn out that this problem is also part of P.

4.3.1 Preliminary results

The main focus of this section will be to prove an extension of lemma 16. Furthermore a few differences with section 3.3 will be pointed out.

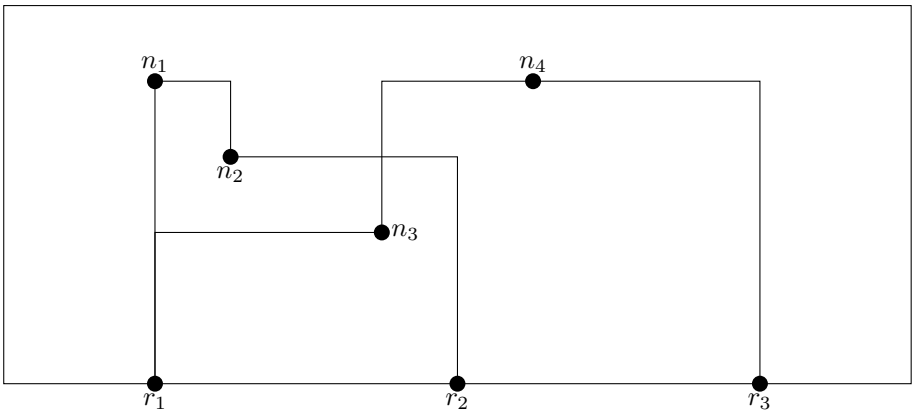
The first difference with section 3.3 is that, in this specific problem, all reservoirs will be used, unless there are less than three customers. If there are only one or two customers, then it is very easy to construct an optimal solution and an algorithm wouldn't even be necessary. If there are more than two customer, then there always exists an optimal solution that uses all reservoirs. The proof of this is the same as the proof that was given in lemma 15.

Some similarities occur when one takes a look at example 8. In this example a switch goes from reservoir 1 to reservoir 3 and from reservoir 3 to reservoir 2. One should notice that in the current problem this situation still occurs, since the solution given also indicates the optimal matching for this problem. It is therefore still possible that an optimal solution will switch to a reservoir that is not directly next to the one it comes from. The algorithm should account for this and it will be shown that it indeed does.

One should note that situations like example 5 and 6 do not occur within the current problem. In example 5 this is due to the fact that the algorithm will try to put the reservoirs in such a manner that there is always at least one point close to it. The situation in example 6 will not occur, mainly because this problem uses the L_1 metric. It would not matter for the edge from n_2 to r_3 to first visit r_2 or not. This is not exactly seen in the example given in 3.3, but in the current problem, the location of r_2 would be slightly different and directly beneath n_2. In that case it is easy to see that it does indeed not matter to first go to r_2 instead of going to r_3 directly.

At this point, there are no further remarks to be made and it is time to extend lemma 16. This is however not easy to do. The current problem has a lot of similarities with lemma 16 and all cases that are handled in lemma 16 will come back. They can all be proved the same way as in the previous section and this is not hard. The hard part is that with three reservoirs, there is now a possibility that the following situation occurs.

Example 12.



This exception is hard to solve. First of all, one would need to assume that there are a lot of customers really close to each reservoir that are not drawn in the picture. If this was not the case, then the solution would never be optimal, because the reservoirs are not in their optimal location. After this is assumed, one can solve this by connecting both n_2 and n_3 to r_2. This solves the problem only if $n_3 \in N_{r_2}$. One could assume this is not the case, then the problem can be solved by connecting n_2 to r_1 and connecting n_4 to r_2 and making a switch between r_1 and r_2. This would never be an optimal solution, but it would be cheaper if $n_4 \in N_{r_2}$. If one also assumes that this is not the case then one could argue that n_3 and n_4 could both be connected to the assumed customers close to r_2, because they now have to lie really far away from each other. This would surely make the solution cheaper again if the customers lie on appropriate heights.

The problem with the extension of lemma 16 is that there are an enormous amount of exceptions that in principle should all be solved, but it hard and tedious work to solve every exception or even write them all down. Therefore, the lemma will not be extended. However, due to the nature of the L_1 metric, it is assumed that every situation can be solved in some way.

One should take a look at the possible single tours in an optimal solution. By now, the reader should have some insight in this and should notice that the only possibilities for single tours are the same ones that were already described in section 3 of chapter 3. With that, it is now possible to construct an algorithm.

4.3.2 Constructing the algorithm

Lemma 16 was not extended, but it was assumed that it could be extended when all possibilities would have been written down. The algorithm that will be constructed assumes that it can be extended and therefore uses the same approach as section 2 of chapter 4.

One should take care that at this point, when the pseudocode would have been written as in chapter 3, then the algorithm would become immense. This is therefore not done and the current pseudocode just shows the steps that should be taken at what point in time.

Input: A set N of customers
Initialize: An empty ordered set P

Order all customers of N according to x coordinates
Check if $|N|$ is even or odd

Set: $N_L = \{n_1\}, N_M = \{n_2\}, N_R = N - \{n_1, n_2\}$

while $N_L \neq N$ **do**
  while $N_M \neq N - N_L$ **do**
    if $|N|$ is even **then**
      for all Possible pairs of switches **do**
        for all Possible pairs single tours **do**
          Find the optimal matching of N without these points
          Add all resulting tours to P
          Check if P is cheaper than the previous solution and save it if it is
          Do the same without the single tours.
          Do the same with one single tour as a single tour switch
        end for
      end for
    else if $|N|$ is odd **then**
      for all Possible pairs of switches **do**
        for all Possible triple of single tours **do**
          Find the optimal matching of N without these points
          Add all resulting tours to P
          Check if P is cheaper than the previous solution and save it if it is
          Do the same for every one of the single tours as only single tour
          Do the same for every one of the single tours as a single tour switch
        end for
      end for
    end if
    Add the first point of N_R to N_M
  end while
  Add the first point of N_M to N_L
  Set N_M to be the first next customer that is not an element of N_L
  Set $N_R = N - (N_L \cup N_M)$
end while
Output: An ordered set P, which will be an optimal path

4.3.3 Proving optimality

Again, it is assumed that lemma 16 can be extended. Without this, the algorithm would not be correct, because there could exist scenario in which the optimal solutions do not have this form. When it is assumed, the proof that the algorithm is correct is the same as the proof of section 2 of chapter 4.

First of all, there are always $O(|N|^3)$ possible subsets. The algorithm has to do a bit of work for every one of them. If the number of customers is even, then there are $O(|N|^2)$ possible single tours and there are $O(|N|^4)$ possible switches. When these are chosen, the algorithm will find an optimal matching, which will take $O(|N|^5)$. Therefore, the algorithm will find an optimal solution in $O(|N|^{14})$. If there is an odd number of customers, then there are $O(|N|^3)$ possible triples of single tours. There are also $O(|N|)$ possibilities for single tours and single tour switches, but this extra work gets dominated by the time it takes to solve for the triples. After that there are $O(|N|^4)$ possible switches and then the algorithm again has to find an optimal solution in $O(|N|^5)$ time. The total time is $O(|N|^{15})$. The worst case scenario therefore takes $O(|N|^{15})$. Which is still polynomial and therefore the problem finds itself in P.

4.4 k reservoirs, L_1 metric

The next logical step would be to look at a general amount of reservoirs and try to find an algorithm that solves this problem to optimality. The previous section might give an idea that this problem is also solvable in polynomial time with the same kind of algorithm. Again, this requires lemma 16 to be further expanded on and again it is highly expected that this is possible, but one would have to solve a lot of different cases. This might however become easier after $k = 4$, since any possible scenario with two tours can be in some way reduced to a scenario with only four reservoirs. Adding more reservoirs in between might make some situations even easier to solve. The nature of the proof of lemma 16 gives a slight indication that if one proves the extension of lemma 16 for $k = 4$, that it will hold for all k. Still, one has to prove every possible case and there are a lot.

However, it is strongly suggested that for general k, the lemma still holds and therefore the algorithm can always be modified to solve the problem to optimality. One should then look at the running time of these algorithms. With k reservoirs the number of possible subset will be of $O(|N|^k)$. After that the algorithm still has to try to find solutions with single tours. If there are k reservoirs, then the amount of single tours also increases, as was already stated in section 4 of chapter 3. Also, the algorithm has to force $k-1$ switches and it has to force every possible tuple of $k-1$ switches. After that the algorithm will be able to find an optimal solution, but this will be done in at least $O(|N|^k)$ time. If k is part of the input, then this algorithm will have a running time that is exponential in the input. It is therefore not polynomial anymore and it can not be concluded that the problem in this case is in P. However, when k is fixed, the running time will only be polynomial in $|N|$. The algorithm will not be very fast, because three reservoirs already gave a running time of $O(|N|^{15})$, but it will always be polynomial, which means that it is part of P.

5 Free reservoirs, Euclidean distances

These collection of problems is very similar to the first set of problems that were analyzed. The only difference being that the location of the reservoir can be chosen. Finding the optimal location will turn out to be a very tedious job. It will turn out that this is actually difficult to do in reasonable time or with an exact result. The first section will therefore already not yield any meaningful results and a continuation would not be meaningful either.

5.1 One reservoir, two components

This section will look at the problem with one reservoir. The first thing that has to be done is to analyze the optimal location of the reservoir.

5.1.1 Choosing the optimal location

First of all it should be noted that when there is an odd number of customers, then the optimal location is dependent on the matching that is being used, since the single tour that will exists will have two edges to the reservoir, which means that it will have more weight when finding an optimal location.
This problem also occurred in chapter 4, but it was proven that in that situation it did not matter, because the optimal location did not change when adding one customer.

The distance function between a customer $n_i = (x_i, y_i)$ and the reservoir $r_1 = (x, 0)$ is

$$f_i(x) = \sqrt{(x_i - x)^2 + y_i^2}.$$

In the previous chapter, it was possible to ignore the y values, because they all added up to some constant, independent of x. This is sadly no longer the case and all y values need to be accounted for. The function that has to be optimized will then be

$$f(x) = \sum_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} \sqrt{(x_i - x)^2 + y_i^2},$$

where $n = |N|$. One should note that every f_i is logically a convex function, which means that f is a sum of convex functions and therefore also convex.

When optimizing a convex function, one could use iterative functions. However, these functions do not yield very good results in some bad cases and it can not be assured that an optimal solution can be found in polynomial time. Only nice functions can really guarantee results in reasonable time. The function that has to be optimized is however really difficult, because of all the square roots. An exact result in reasonable time can therefore not be guaranteed.

5.1.2 Preliminary results

One should still notice that single tours still occur in some optimal solutions. Once again, this can only happen when the number of customers is odd and there will be only one single tour in that case. If there are more then one could combine them again to make a cheaper solution.

One should note however that the optimal location of the reservoir is dependent on which of the customers will be the single tour. In section 1 of chapter 4, this was not a problem, because

adding one dummy customer would not change the optimal location of the reservoir. However, when adding one dummy customer to simulate the single tour, the optimal location does change. Therefore the algorithm will need to calculate the optimal location every single time for every different single tour.

5.1.3 Algorithm

The algorithm will try to construct a path P, just like in previous chapters.
When there is an even number of customers, there will be no single tours. The algorithm will try to find an optimal location for the reservoir and will account every customer exactly once. Then it will find an optimal matching and combines this to construct an optimal path. When there is an odd number of customers, then the algorithm will follow the same procedure, but when finding the optimal location, it will always count one of the customers twice. This will be the single tour. Then it will find an optimal solution for that single tour and finally it will find the cheapest one of those solutions.

Input: A set N of customers and a location for r_1
initialize: An empty ordered set P

Determine if $|N|$ is odd or even

if $|N|$ even **then**
  Find the optimal location for the reservoir
  Use Blossom V to find the optimal matching M of N
  Number edges $e_1, \ldots, e_{\frac{|N|}{2}}$ in M
  for $i = 1, \ldots, \frac{|N|}{2}$ **do**
    Add $(r, v_i), (v_i, w_i), (w_i, r)$ to P, where $e_i = (v_i, w_i)$
  end for

else if $|N|$ is odd **then**
  for all $n \in N$ **do**
    $P = \emptyset$
    Use n as a single tour and find the optimal location for the reservoir
    Blossom V: Find optimal matching M of $N - \{n\}$
    for $i = 1, \ldots, \frac{|N|}{2}$ **do**
      Add $(r, v_i), (v_i, w_i), (w_i, r)$ to P, where $e_i = (v_i, w_i)$
    end for
    Add $(r, n), (n, r)$ at the end of P
    Check if the current solution is more optimal then previous optimal solution and save it
  end for
end if
Output: An ordered set P, which will be an optimal path.

5.1.4 Proving optimality

The only difference with chapter 3 is the fact that the optimal location now has to be chosen. As was said before, one could use the field of convex optimization to do this, but this might not yield exact results or results in reasonable time. This algorithm will however find an optimal solution when the reservoir is not considered, because it is the same algorithm that was used in chapter 3.
When the number of customers is even, the algorithm needs to find the optimal location. This could be done using convex optimization, but the results may vary when considering running time. Then it needs to find the optimal matching, which will take $O(|N|^5)$ time. The running time really depends on the time to find an optimal location. Therefore no real running time can be given. When the number of customers is odd, the same case occurs. One should now consider all possible single tours, but still the running time depends a lot on finding an optimal location of the reservoir. Since this running time can not be given, it can not be concluded that this problem is in P or not.

6 Several colors

The next section will be dedicated to a different variation of the problem that was analyzed in section 3. In this variation, it is possible that there are several different components and every customer will need a specific kind of component. This will be visualized by giving the components different colors which will just be named 1,2,3 and so on.

This problem would not be very difficult if the different components would all reside in the same reservoir and the arm could just get any component it needs for its next tour. It will therefore be assumed that every reservoir can only hold one specific color of a component. Furthermore it will be assumed that every component of a specific color i has a different set of reservoirs R_i, which might differ in size.

This problem has several applications in real life, because it will occur that a chip needs different components and holding more then one kind of component in a reservoir makes it difficult for a machine to find the right components. Solving this problem makes it therefore easier for this kind of real life problem to be solved efficiently.

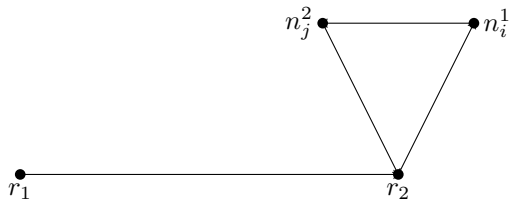
Again, it is possible to fix the location of the reservoirs or to allow choosing the location. This was done in the last problems too. Since solving the problem without fixed reservoirs was very difficult, it will be assumed that the locations are always fixed with this problem. Also, it is assumed that the location of the reservoirs do not overlap, because if this would happen, this would make the problem less interesting.

6.1 Two colors, Two fixed reservoirs, free movement

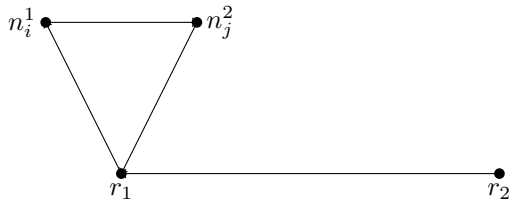
It will be assumed that the reservoirs are not located at the same location. If this would happen, the problem would be reduced to one color, with one fixed reservoir and free movement. This problem is the same as discussed in chapter 3. Furthermore the customers will again be numbered from left to right, but there will be made a difference between customers of a different color. The customers of color i will be numbered $n_1^i, \ldots, n_k^i$.

A first intuition to solve this problem would be to use an optimal matching of the customers again. Trying this will result in the first major problem. It might be possible that in an optimal matching, two customers of different colors are matched together. If the mechanical arm wants to use this tour, it must first visit both reservoir to get the components required. This will majorly increase the cost of this tour and if one would account for this, the optimal matching that was found, might not be optimal anymore. At first, try to account for this by altering the distances between two customers. This has to be done carefully, because there are a few possible ways to match two customers of a different color.

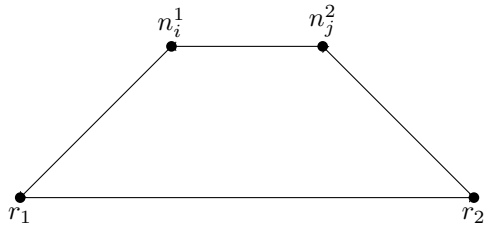
Example 13.
In this first example, the arm starts in r_1 and goes first to r_2 to get the necessary components. The tour ends in r_2 and this will in fact create a switch from r_1 to r_2.



Example 14. *The second example is similar to the first. Instead, a switch from r_2 to r_1 is created.*



Example 15. *The last category is a tour that can start in either r_1 or r_2 and will then return to the original reservoir.*



It must be noted that in previous chapters the direction of edges could be changed. This was done in many proofs and it works, because switching the direction will not hurt the feasibility of the solution after some more alterations. In this problem, the direction of the first two switches can not be changed without increasing the cost. The arm needs to start in a specific reservoir to get the required components. It can not start in the other reservoir and simple walk the tour backwards, because it would miss components that it needs for some of the customers.

Furthermore, it must be noted that not every tour can be started from every reservoir, when the arm is currently in a reservoir of color 2 and it wants to visit two customers of color 1 next, then it has to visit a reservoir of color 1 first. In the original problem the customers where indistinct, but this is no longer the case. If one still wants to make an optimal matching, then one should account for this when looking at the distances.

One should take great caution with these examples. If some of the locations of the customers are shifted slightly, it would have been cheaper to make two single tours. One of the single tours would then have to be a single tour switch, which might be counter-intuitive. However, since these customers are of different colors it is possible that this is the cheapest solution. This becomes more problematic with example 13, because this will now become a switch instead of just a tour attached to reservoir 1.

The distance between two customers will be defined as follows. If the two customers have the same color, the distance will be defined as the Euclidean distance between the customers plus

the shortest Euclidean distance from one of the customers to one of the reservoirs of the appropriate color plus the shortest distance from the remaining customer to an arbitrary reservoir. One should note that this is always the shortest distance to service these two customers together.

If the two customers are of different colors, then the distance will be defined as the minimum of two distances. The first distance is the Euclidean distance between these customers plus the shortest distances between one of the customers to his closest reservoir plus the distance between the two reservoirs. The second distance is the shortest distance between these customers to their appropriate reservoir plus a sum of distances such that there will be at least one single tour switch. This might sounds complicated, but this distance will essentially again be the shortest distance to serve these customers together.

When finding an optimal matching with these distances, one essentially matches customers together such that their paired contributions are minimal. This might be in the form of an edge between them or in special cases as two single tours. It is still possible, that when an optimal matching is found, this optimal matching does not have a switch yet. This would not be feasible, but can be accounted for in an algorithm. This is done by forcing a switch and optimizing all other customers. This leads to some problems.

The first problem is that a switch might not be needed at all. This might occur, when for example all customers only have one color. One should determine certain requirements for a solution to need a switch.

The second problem would be that if a solution has several switches, the direction of the switches does not provide a feasible solution. This could happen, if all switches go from r_2 to r_1. There would be no way to go back to r_2 and therefore there would be no way to service the remaining switches. This problem occurs only in this chapter, because in previous chapters the direction of switches could be switched. In this chapter, this is not always possible and therefore it is possible that this situation occurs.

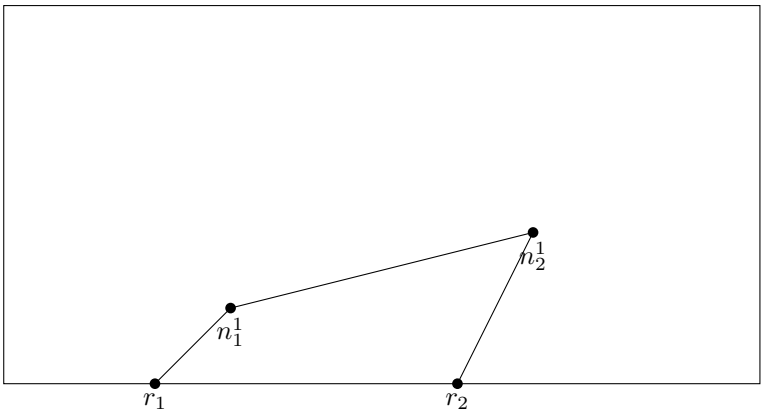
Furthermore, one should take a look at the single tours. With two reservoirs and two different colors, new possibilities occur for single tours.

The described problems will be discussed in the next sections.

6.1.1 Necessity of a switch

First of all, one might expect that if all customers have the same color, only one reservoir has to be used. This could be due to the fact that all tours have to start in the same reservoir, because the arm needs to get the right components. However, it is possible that the last tour ends in the other reservoir to make the solution cheaper. An example of this will be shown next.

Example 16.



This problem is easily solved. If all customers have the same color, assume color 1, then one could look at the neighborhood of r_2. If there is any customer close then there always exists an optimal solution that uses r_2. This can be easily proven with a simple exchange argument.

It is also possible that there is no switch needed, when all customers can be serviced by tours of the form of example 13. There might be some tours that are connected to one reservoir only, but this would still generate a possible optimal solution that does not have a real switch. One could try to account for this by trying to find a solution without forcing a switch. An algorithm would then have to check if the found solution is feasible.

6.1.2 Direction of switches

The first remark that must be made is that not only the switches of example 13 and 14 can not be reversed. All switches can not be reversed without increasing the cost dramatically. This is because any other switch would be a switch from one reservoir through one or two customers of the same color to the other reservoir. Since the other reservoir is not the same color, the switch can not be reversed without visiting the first reservoir first. This will of course increase the cost and therefore it can not be reversed mindlessly.

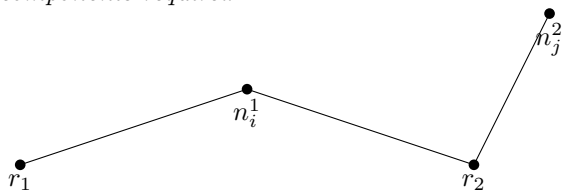
This is an important thing to notice. In previous chapters, the algorithm tried to force a switch and then find the cheapest solution for that switch. Since it was known that there would always be a switch, at some point the optimal solution would be found. This worked, because one could be sure that the algorithm would have enough switches and therefore the solution that was found would be optimal. One would like to try the same style of algorithm for this problem. However, if one tries to force a switch and tries to find the cheapest solution, it might occur that there arise a number of switches that go in the same direction and can not be reversed. Therefore a solution found in the way that a solution was found in previous chapters might not be feasible.

6.1.3 Single tours

In previous chapters, it was proven that every reservoir can have at most one single tour attached to it. The first remark of this section is that this is not true anymore if there are several colors. This will be shown in the next example.

Example 17.
This example shows that it is possible that a single tour starts in r_1 and visits one customer

of color 1. Then it goes to r_2 and finally serves the customer of color 2. This is cheaper then servicing both customers in the same tour, because then the arm first has to visit r_2 to get the components required.



This shows that one should look with caution at the possibilities of single tours, but one should also note, that this can only happen if the single tours are of two different colors. If there are two single tours attached to a reservoir of the same color, then they can still be combined to form one single tour. All possible single tours that can occur now are listed below. One could find examples of all these cases and one can also check that these are all possible cases and no other exists.

- A single tour switch and a single tour, connected to the end of the single tour switch if $|N|$ is even;
- Two single tours, connected to different reservoirs, if $|N|$ is even;
- No single tours at all, if $|N|$ is even;
- A single tour switch if $|N|$ is odd;
- A single tour, if $|N|$ is odd.

One should still account for these single tours and should match the remaining customers in some way to optimality.

6.1.4 Summary

When considering only two colors, the problem already becomes immensely complicated. This section will be dedicated to summarizing the findings of the previous sections.

First one should list all possible matchings that could occur between two customers:

- Two customers of color i:
  - Switch from r_i to r_j if at least one customer lies close to r_j;
  - A regular tour from r_i to r_i.
- Two customers of different colors:
  - Switch from r_1 to r_2, if both customers lie close to r_2;
  - Switch from r_2 to r_1, if both customers lie close to r_1;
  - Tour from either reservoir to itself, like in example 13.

Also it is of course possible to force specific tours that are not as cheap as possible. For example, one could force a switch from r_1 to r_2 via two customers of color 1 that are both close to r_1. It would be cheaper to connect them both to r_1, but it might be optimal to force a switch out of

them.

One should also note that the direction of the switches do matter. One can not change the direction of any switch.

Secondly one should list all possible single tours and single tour switches that could occur:

- One single tour switch and one single tour, connected to the end reservoir of the single tour switch if $|N|$ is even;
- Two single tours, connected to different reservoirs, if $|N|$ is even;
- One single tour switch if $|N|$ is odd;
- One single tour, if $|N|$ is odd.

In general, both reservoirs will be used in some way. Either by connecting complete tours to them, or by passing through them like in example 15. Once again, one should account for single tours. This also means that apparently the remaining customers must be matched together. This can be proven the same way as one did in chapter 3.

Now one has to find a way to notice if the optimal solution needs a switch. This seems a difficult and rather random task, since the tours of example 15 could be used over and over again, without the use of a proper switch. A way to determine if one actually needs a switch seems to be difficult to nail down too.

After trying to force some switch, this still remains difficult, because the customers have to be matched in some way. Using Blossom V could result in a matching in which there arise unwanted switches. It seems therefore that Blossom V can not be used recklessly and this would imply that a new method of finding an optimal matching should be deduced.

Sadly, no conclusion can be given about this problem, except for the remarks that were made above.

7 Summary

In the first chapter, the problem with k fixed reservoirs and C components and a free-moving arm was analyzed. For $C = 2$ and $k < 3$ it was shown that there exists an algorithm that solves this problem to optimality. The running time of these algorithms was polynomial and therefore the problems found themselves in P. If $k \geq 2$ it was shown that there occur a lot of different situations that are difficult to solve. An algorithm was given that solves this problem to optimality. If $C > 2$ it was shown that there does not exist an algorithm that solves this problem in polynomial time. An article was recommended to give more details about these proofs.

In the second chapter, the problem with k free reservoirs and C components and an arm that was moving according to a L_1-metric was analyzed. If $C = 2$ and k is a fixed number, then it was proven that there exists an algorithm that solves this problem to optimality. The running time of these algorithms was also given and it turned out that for $k = 1, 2$ the running time was polynomial. If $k > 2$ the running time was at least $O(|N|^k)$. For any fixed k, the running time would have been polynomial. However, if k was considered part of the input, the running time would be exponential in k. This algorithm could therefore not give a conclusion about this problem being in P or not.

In the third section the first problem was revisited with free reservoirs. It was shown that one has to optimize a function. After that, the problem becomes like a combination of the first and second chapter and can be solved. Finding the optimal location was however difficult and one could not guarantee exact results or results in reasonable time.

In the last chapter, it was analyzed what happens if there are different components. It turned out that with even as few as two different components this problem becomes immensely complicated. A few suggestions were given about what an algorithm should account for and what different situations could occur.

References

[1] Vladimir Kolmogorov, *Blossom V: a new implementation of a minimum cost perfect matching algorithm*, Springer and Mathematical Programming Society, 2009.

[2] Barrie M. Baker, M.A. Ayechew, *Agenetic algorithm for the vehicle routing problem*, Computers Operations Research 30, 787 – 800, 2003.

[3] Kaarthik Sundar, Sivakumar Rathinam, *Generalized multiple depot traveling salesmen problem—Polyhedral study and exact algorithm*, Computers Operations Research 70, 39-55, 2016.

[4] Benoit Crevier, Jean-Fran¸cois Cordeau and Gilbert Laporte, *The Multi-Depot Vehicle Routing Problem with Inter-Depot Routes*, 2005.

[5] Jean-François Cordeau, Gilbert Laporte, Martin W.P. Savelsbergh, Daniele Vigo *Vehicle routing*, Handbook in OR MS, Vol. 14, 2007.

[6] Tetsuo Asano, Naoki Katoh, Hisao Tamaki, Takeshi Tokuyama, *Covering points in the plane by k-tours towards a polynomial time approximation scheme for general k*, Dept. of Engr. Informatics, Osaka Electro-Communication University, Neyagawa, 1997.

[7] Alexander Schrijver, *A Course in Combinatorial Optimization*, 2017.

[8] M. Khachay, H. Zaytseva, *Polynomial Time Approximation Scheme for Single-Depot Euclidean Capacitated Vehicle Routing Problem* Springer International Publishing Switzerland, 2015.