# Bus Routes for ALCAN Iceland

Einar Leif Nielsen s041910

December 3, 2006

Supervisor: Professor Jesper Larsen

Informatics and Mathematical Modelling
Technical University of Denmark

# Abstract

In this thesis a problem, presented by ALCAN Iceland, is put forth. The problem, called the bus route problem, examines the pickup of employees on route to an aluminium plant. Therefore a depot is defined and also a set of pickup locations. A bus must navigate through the points choosing only the most important locations. The problem is presented mathematically and a meta-heuristic, simulated annealing, is used to solve the problem. The are a number of tests put forth. A good cooling schedule is calculated. A matrix determining the probability of a neighborhood is constructed. The best method of node insertion into the solution is found. The algorithm calculated structured solutions provided with non-randomly generated data sets. The simulated annealing algorithm was then vompared to a GAMS program, returning values between the upperbound and the objective value calculated with GAMS. Comparison between tabu search algorithm for TOP and the simulated annealing algorithm showed that the former is faster for small data sets and nearly always returns better objective values. Finally a constraint forcing a bus to travel for a ceratain amount of time before stopping again was implemented.

# Acknowledgements

I would like to thank Professor Jesper Larsen my supervisor on the project for always lending a helping hand when needed and Professor Jens Clausen for filling his shoes when Jesper was absent. I would also like to thank Min Wen who assisted me in any way that was requested, among other things she contributed greatly to the GAMS program, discovered how to make the model linear and added the distance constraint to the model. Snorri Páll Sigurdsson, my office neighbour, who helped in numerus ways and inspected my report. Professor Páll Jenson was instrumental in the project as he was the one who found the problem and handed it to me. He was also the project supervisor for the University of Iceland. At ALCAN I would especially like to thank Þorsteinn Ingi Magnússon who was the project supervisor for the company, he got me all the data that ALCAN supplied and a desk at there office when that was needed. Also I would like to thank Arngrímur Einarsson and Fannar Ørn Thordarson for proof reading the thesis.

# Contents

# Chapter 1

# Introduction

## 1.1 Status Description and Motivation

This exam project is done by Einar Leif Nielsen for ALCAN in Straumsvík, Iceland. Main supervisor on this project was Jesper Larsen, at IMM DTU. Co supervisors were Min Wen, PhD student at IMM DTU, and Páll Jenson, professor at the University of Iceland. In the middle of the project Jesper Larsen had to leave for New Zealand, for a few months, in the meantime Professor Jens Clausen took over Jespers duties on the project.

This project deals with the pickup of employees for the ALCAN aluminium plant at Straumsvík Iceland. ALCAN is the second largest aluminium manufacturer in the world. Its name is derived from the words ALuminium and CANanda. It has over 470 facilities in 55 countries[1]. The aluminium factory in Iceland is the 11th [5], largest, in the corporation. It is located in Straumsvík, which is a just out side of Hafnarfjördur, a suburb of Reykjavík. Aluminium oxide is imported from Australia and manufactured into aluminium. The metal is then transported overseas for further work. ALCAN Iceland employees a around 470 persons [5] and was the first aluminium plant constructed in Iceland. There are now three and more are planned.

ALCAN's bus system, which picks up employees, was first taken into use 30 years ago. That system has since grown and new pickup points have been added without calculating their location. Now the system is very complicated and has grown very expensive. A newly implemented tax on diesel fuel, by the Icelandic government, has lincreased the cost even more. Therefore ALCAN decided to see if a more economical method for picking up employees exists.

The goal of this project is to find more economical bus routes and to see if the number of buses, and thereby routes, can be decreased. ALCAN hopes to decrease the cost of the system by at least 10-15%. This can be achieved by inspecting the location of pickup points to see if all current pickup points are necessary. Also new pickup points can be introduced that would be better located than those currently in use. New areas[2] will not be added to the current system. ALCAN also hopes that this will decrease travel time for the employees as some employees spend nearly an hour on the bus.

[1] http://en.wikipedia.org/wiki/Alcan
[2] Neighborhoods and suburbs.

ALCAN prefers a general solution as it receives a large work force during the summer months that relieve other employees during summer vacations. This does not mean that special solutions should be excluded. They will be looked into and their importance estimated. ALCAN has requested that various types of solutions are to be inspected.

The current number of employees at ALCAN is 467 divided on three shifts. A day shift, 08-16; an afternoon shift, 16-00; and a night shift, 00-08. There are also three types of employees. Those who work the day shift, those who work the day shift and the afternoon shift and those who work on all three shifts. Also there are those who work weekends and those who do not. As is the case with most workplaces, that use a shift system, at no time is all the workforce present at the plant. So the largest work force is present during the day shift on the weekdays while the smallest workforce is present during night time on the weekends.

For the day shift pickup ALCAN uses 68 pickup points and approximately another 15 are added for the night and afternoon pickup (during these times some of the other pickup points are excluded). New pick up point will be added to the system, these points can be local bus stops or other strategically chosen points, while others will be removed. The current routes are of different lengths the longest taking approximately 52 minutes, driven in the morning during weekends and holidays; and the shortest approximately 29 minutes, driven in the morning on weekdays.

Currently Hópbílar supply the buses used in picking up employees for the aluminium plant in Straumsvík. Hópbílar is a privat bus company and one of the biggest in its field in Iceland. They have served ALCAN well and both companies want to continue that cooperation.
Other transportation possibilities mentioned in this report are: the local bus system, run by a company called Strætó; and the local taxi services, which are many.

## 1.2 Outline of the Thesis

There are five chapters in this thesis excluding the first, this one. Each of these five chapeters examin a different part of the problem that has been introduced.

The second chapter annalysis the problem presented, defines it and presents a mathematical model. There is also a review of relevant problems. The final section in the first chapter examins methods that have been implemented on similar problems and their results.

The third chapter looks at the theory of the algorithm used for solving the problem. Also implementation of this algorithm is explaind.

The fourth chapter looks at various tests, or compuational experiments. In this chapter the best parameters are calculated and implemented. Various data sets were generated and other data sets obtained.

The fifth and final chapter summarises the conclusions reached in this project. The fifth chapter also posses questions regarding further work, such as: Is there any further development of the problem, or method, possible? Are results useful and other questions?

There is also a large appendix in this report containing various results from experiments, analysis and the algortihm used.

# Chapter 2

# Bus Route Problem

In this section a number of different solutions for solving this problem, are explored. ALCAN's bus route problem will from now on simlpy be reffered to as the bus route problem.

## 2.1 Analysis of the Realistic Possibilities

For this problem there are many possible solutions, these types of solutions have been categorized in two types.

1. *First type of solutions* only rely on one transportation possibility. That is only one company will transport employees to and from the aluminium plant.

2. *The second type of solutions* are combinations of two transportation possibilities.

It was considered and rejected to use combinations of more than two transportation possibilites. The reason for this is that solutions of the second type covered all hours of the day, 365 days a year. Therefore a new more complex combinations would offer no improvement over the solutions of type 2. Also combinations of more than two transportation possibilities are likely to be too expensive.

In total there are 8 solutions of the first type and 25 combined solutions. Solutions of type one are defined in table 2.2.

### 2.1.1 SWOT Analysis

Strength, weakness, opportunity and threats analysis, or SWOT analysis, was used to determine which solution would be best suited in solving the bus route problem. This is a method often used to define the pros and cons. In this method:

**strength** represents helpful internal factors
**weakness** represents harmful internal factors
**opertunities** represents helpful external factors
**threats** represents harmful external factors

When using SWOT analysis one has to define internal and external factors. In the case of the bus route problem **internal** factors were defined as: The author of the project, ALCAN and the project supervisors.
**External** factors were defined as: The employees of ALCAN, transportation companies and the general public of the greater Reykjavík area.
After internal and external factor have been defined a table is constructed. An example of a SWOT analysis table can be seen in Table 2.1.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general solution that takes into account employee turnover. Works all year round, 24 hours a day. This solution is not too simple to be considered a exam project. | Not ALCAN's desired solution. In this solution new nodes without a predefined location cannot be used. Hard to estimate the general population of an area. |
| **External** | Decreases travel time. | Decreases profit for Hópbílar. Decreases the current amount of service provided by ALCAN. |

Table 2.1: This table show how solution of type 1 was analysed. SWOT analysis of all solutions can be viewed in appendix $B$.

### 2.1.2 Combined Solutions

All possible combined solutions are shown in table 2.3. Although due to the number of possible combinations more information could not be included in the table. Therefore a short explination, of Table 2.3, is in order for example look at combination 1(Combo 1). This is a combination of solutions of type 4 and type 1, both are defined in table 2.2. This combination proposes the use of Strætó, the local bus system, when possible and Hóbílar, a private bus company, when Strætó is closed. This is useful as the local bus system is closed during night time and during holidays such as Christmas.

A different combination type, is combination 15(Combo 15). This combination uses solutions of type 2 and 5. Note though that the combination uses an extreme solution of type 2. An extreme solution tries to limit the number of routes and travel time of a single route as much as possible. So the solution would provide a few pickup points were Hópbílar would stop. Employees however would have to get to these pickup points by themselves.

In appendix $B$ all combination solutions are defined and analyzed with the SWOT method.

Table 2.2: Possible solutions for the problem

| Name | Description | Transportation |
|---|---|---|
| **Type 1** | Use current pickup points along with new ones (predefined, such as local bus stops). Estimate the importance of each pickup point by the number of people living close to it, the amount of parking and connection to local transit system. *Buses* from Hópbílar are used to pick up employees. | Hópbílar |
| **Type 2** | Same as type 1 except importance of pickup points is decided by the number of employees that live close to them. *Buses* | Hópbílar |
| **Type 3** | Same as type 2 except a software, such as ShorTrec from AGR hf., is used to determine the bus routes. A new route can be calculated as often as ALCAN desires. *Buses* | Hópbílar |
| **Type 4** | Uses the *local bus system*, buses, to pickup employees and return them. | Strætó |
| **Type 5** | *Car pooling*. Each car will be given a driving diary and receive a payment for gas used at the end of the month. It would be necessary to write a program that would put five people together as a part of a car pooling team. | Employees |
| **Type 6** | *Driving grant*. Each employee would receive an increase in pay to compensate for the lack of buses. The employees would then drive themselves to work. | Employees |
| **Type 7** | *Car pooling with taxis*. A taxi would pickup employees and return them. Each taxi would be filled with passengers. A program would tell the taxi service where and when to pick up an employee. | Taxi service |
| **Type 8** | Same as type 1 except the pickup points would be calculated so that there location was good and not from predetermined points. *Buses* | Hópbílar |

Table 2.3: Possible solutions for the problem

| Name | Description | Transportation |
|---|---|---|
| **Combo 1** | Type 4 and type 1. | Hópbílar and Strætó |
| **Combo 2** | Type 2 and type 4. | Hópbílar and Strætó |
| **Combo 3** | Type 3 and type 4. | Hópbílar and Strætó |
| **Combo 4** | Type 5 and type 4. | Employees and Strætó |
| **Combo 5** | Type 6 and type 4. | Employees and Strætó |
| **Combo 6** | Type 7 and type 4. | Taxi service and Strætó |
| **Combo 7** | Type 8 and type 4. | Hópbílar and Strætó |
| **Combo 8** | Type 5 and type 6. | Hópbílar and Strætó |
| **Combo 9** | Type 7 and type 6. | Hópbílar and Strætó |
| **Combo 10** | Extreme solution using type 1 and then use type 4. | Hópbílar and Strætó |
| **Combo 11** | Extreme solution using type 2 and then use type 4. | Hópbílar and Strætó |
| **Combo 12** | Extreme solution using type 3 and then use type 4. | Hópbílar and Strætó |
| **Combo 13** | Extreme solution using type 8 and then use type 4. | Hópbílar and Strætó |
| **Combo 14** | Extreme solution using type 1 and then use type 5. | Hópbílar and employees |
| **Combo 15** | Extreme solution using type 2 and then use type 5. | Hópbílar and employees |
| **Combo 16** | Extreme solution using type 3 and then use type 5. | Employees and Hópbílar |
| **Combo 17** | Extreme solution using type 8 and then use type 5. | Employees and Hópbílar |
| **Combo 18** | Extreme solution using type 1 and then use type 6. | Employees and Hópbílar |
| **Combo 19** | Extreme solution using type 2 and then use type 6. | Employees and Hópbílar |
| **Combo 20** | Extreme solution using type 3 and then use type 6. | Employees and Hópbílar |
| **Combo 21** | Extreme solution using type 8 and then use type 6. | Employees and Hópbílar |
| **Combo 22** | Extreme solution using type 1 and then use type 7. | Taxi service and Hópbílar |
| **Combo 23** | Extreme solution using type 2 and then use type 7. | Taxi service and Hópbílar |
| **Combo 24** | Extreme solution using type 3 and then use type 7. | Taxi service and Hópbílar |
| **Combo 25** | Extreme solution using type 8 and then use type 7. | Taxi service and Hópbílar |

### 2.1.3 Chosen Solution

From the SWOT analysis it was determined that solution of type 2 was best suited. The reason for this choice is that this solution is relatively simple to program and therefore a good place to start the project. Also this would provide a solution for ALCAN. Although not as general as they may have preferred but ga ood special solution. The definition of solution type 2 can be seen in table 2.2. Small changes have been made to this solution to better suite the needs of ALCAN. Solution of type 2 was defined as:

> Use current pickup points along with new ones (predefined, such as local bus stops). Estimate the importance of pickup points by the number of employees that live close to them, the amount of parking and connection to local transit system. *Buses* from Hópbílar are used to pick up employees.

## 2.2 Problem Definition and Description

The problem as presented by ALCAN gives a geographical set, a set of employees, a set of buses and a set of locations(pickup points). The aluminium plant also has a predefined location and all buses must finish their route there.

Let us first look at the geographical set. Within this set are the possible locations of pickup points, as ALCAN has defined some areas outside of there routes and they do not intend to increase this area. Therefore new pickup points must be located within the geographical set. The travel between all points in a set is called the travelling salesman problem or TSP. In this problem one must navigate trhough a number of points and then return to the point of origin, via the shortest travel distance.

The set of employees includes all employees at ALCAN. Although some employees live outside of the geographical set and are therefore not relevant to the problem. This set is not very crucial to the problem but can be useful in determining the importance of a single pickup point. Traveling through a set of points each assigned a profit is similar to the price collecting traveling salesman problem, PCTSP. In that problem one must navigate trhough a set of points leaving from a source point and return having collected a minimum number of profit on the way, via the shortest route. Note though that one does not have to visit all points, in the set, in PCTSP.

The set of buses is important as the number of buses currently in use, in the system, cannot be exceeded. The set of buses will from now be reffered to as the set of routes. If a single bus is in use, that bus will be called an active route or a route in use. The capacity of a bus is not important as the number of people working at ALCAN ar not that many. Therfore the capasity of a single bus is unimportant. A problem dealing with more than one route is called a vehicle routing problem or VRP. In VRP one must navigate more than one route leaving from a depot, visiting all points in the set, and the returning again to the source, via the shortes possible routes.

Locations are crucial to this project. The choice of where a bus should stop or not is important in determining the cost of the system. The only factor concerning this set is that the locations be within the geographical set. How to choose a location will depend on how profitable a

location is. To determin this profit the number of employees living within a certain radius, available parking, connection to local transit and other factors can be inspected. To simplify for now we will assume that the importance of a node is determined by the number of people, within the set of employees, that are living inside a certain radius from the location. These locations will now be referred to as nodes. Nodes in use can also be called an active nodes. The are many possibiltiees to add nodes to the excisting set of locations as long as those nodes are within the geographical set. As time is more of an issue than distance the travel time between individual nodes will be inspected, not the distance. This problem, as it has been defined, is very similar to the team orienteering problem, TOP. There a team of moutaineers must navigate, each on his own, though a number of nodes, therby collecting profit. The goal of TOP is to collect as much profit as possible and it is not necessary to visit all nodes in the set. Also in TOP one must return to the point of origin.

Another problem, regarding the nodes, concerns the distance, or travel time, between two nodes. If two nodes are situated very close to one another the may have overlapping profit. This means that some of the people living within a certain radius from node one also live within said radius from node two. Therefore a constraint forcing the bus to travel a certain time before stopping again can be implemented. Another solution regarding this problem would involve not choosing two nodes too close to one another.

It is the wish of ALCAN to decrease the cost of the bus sytem. This can be achived in two ways. First by decreasing the number of routes in use or secondly by decreasing the number of active nodes. These are therefore defined as the two main factor in ALCAN's problem, the problem will from now on be reffered to as the bus route problem. ALCAN's whishes are to limit the number of nodes and/or routes while picking up as many employees as possible. This means that not all nodes have to be visited, only those who are deemed important enough. Also as the buses themselves are not owed by ALCAN, but by an outside contractor, the buses therefore do not have to start at the plant. This means a if a route is used it will originate from the first node it visits and then make its way to the aluminium plant. The open vehicle routing proble, OVRP, is simlar to this. In OVRP one must navigate a number of routes, all leaving from the same depot, though a set of nodes. All nodes must be visited but the routes do not have to return to the depot. The aluminium plant will from here on be referred to as a depot.

By combining certain elements of the methods described one can formulated a mathematical model of the bus route problem. Alternative methods than those previously described can be used to solve the problem. For example one could assign all employees to certain bus stops and then one would add those bus stops to a bus route. If the route is too long one would then decrease the number of bus stops and reassign the employees to fewer pickup points. This would be done as often as necessary. After employees have been assigned to the bus stops the problem becomes a OVRP.

This method will most likely have a shorter calculation period than the bus route problem. It would take into account the capacity of each vehicle and there is no chance that a bus will stop at two points with overlapping profit. On the other hand the bus route problem is more likely to choose the best possible routes, it is a more general solution and might possibly choose to stop at points with small profit. In conclusion these are both good methods but the bus route

problem seems to fit more to the wishes of ALCAN and therefore is a better candidate for solving the problem presented.

To determine the location of nodes a population function for the area can be constructed. This function would map out the most populated areas and the points with the highest population, hot spots, would define nodes and there profit. The reason this will not be done is that constructing a population function of a city is outside the scope of this project, even though it would give a very general solution. Therefore the method of predefined pickup points is deemed better in comparison.

## 2.3 A Mathematical Model for BRP

The problem defined is the bus route problem, BRP, and it has been compared to various methods such as PCTSP, TOP and OVRP. It has been shown that the bus route problem has alot in common with these other problems but is not the same as any of them.

In this model there are a few sets which need to be defined. $L$ is the set of $n$ locations, nodes, where pickup of employees is possible. Not all of these locations have to be visited. $V = L \cup \{0, n+1\}$ is the set of all nodes, $\{0\}$ represents factory out and $\{n+1\}$ represents factory in. Travel time from node 0 to any other node is none, 0. This is because the bus route problem is an open problem, like the OVRP, and it is not necessary for the busses to start there route at the depot, plant. $A$ is the set of arcs between nodes and K is the set of busses, $K = \{1, 2, ..., N\}$.

To construct a model of the bus route problem, three variables have to be defined.

| Name | Description |
|---|---|
| $x_{i,j}^k$ | The arc between $i$ and $j$, equal to 1 if the arc is driven, by bus $k$, else it is equal to 0. |
| $y_i$ | A binary number equal to 1 if node $i$ is visited else it is 0 |
| $s_i^k$ | This is the stopping time for bus $k$ at node $i$. |

The time, $s_i^k$, is defined as the time when bus $k$ stops at node $i$ and is therefore dependant on previous $s_j^k$ if the bus stopped at node $j \in L$. Also there are a few constants that need to be defined before the model is presented. Constants are all represent with the Greek sympols except for the upper.

| Name | Description |
|---|---|
| $\tau_{i,j}$ | The travel time between nodes $i$ and $j$. |
| $\phi_i$ | The profit for stopping at node $i$. |
| $\delta_i$ | Indicates penalty for stopping at any given node, i.e. the time it takes to stop at any give pick up point. In most cases there is no penalty for stopping at the source and sink. |
| $\alpha$ | This is a bonus factor for profits, a bonus received when a pick up point is chosen. |
| $\beta$ | This is a bonus factor for not using a bus. |
| $M$ | An upper time limit is put on each route, so that travel time for a single employee is not greater than this number. |
| $N$ | Maximum number of buses. It is not desired to use more buses than are currently in use. |

Note that $\tau_{0i} = 0$, when $i \in V$, because it is not necessary for a bus to drive from node 0, but it helps to start there when constructing the routes. Profit can be determined by looking at: population of area, number of employees living close to the node, parking, bus stops or commerce in the area. Some or all of these factors will be used when determining the profit of a node. The solution will try to maximize the profit collected, while minimizing the number of buses used. Time will be a constraint rather than part of the objective function, this is also done in TOP.

A profit, of $\beta$ is gained by not using a bus. Therefore when a bus is not used it travels straight form source to sink, $x^k_{0,n+1} = 1$.

The bus route problem might be applicable in other cases. In these other applications some of these constants might be unnecessary, or others might need to be added. This will depend entirely on the problem the model is applied to. Also cost may vary depending on time of day, or if there is a holiday. This is because a cost of using a bus can have many factors. The greatest of these is probably the start up cost for a single bus. Costs can be considered as many things for example maintenance, driver salary and bus company profit. Also in some cases companies may charge for each kilometer or each liter of gasoline used.

### 2.3.1 Objective Function

The objective function for the bus route problem is now put forth.

$$max \quad Y = \alpha \sum_i \phi_i y_i + \beta \sum_k x^k_{0,n+1}$$

There are two factors in the objective function. The first half of the equation shows the profits gained stopping at a certain node and that is then multiplied with a bonus factor. The second is a positive contribution for every bus not used, $x^k_{0,n+1} = 1$ and all other $x^k_{ij} = 0$, that is then multiplied with a bonus factor. The bonus factor represents for excample the cost of a single bus, $\beta$, or the importance of a single profit point, $\alpha$.

### 2.3.2 Constraints

Here are the constraints constructed for the bus route problem.

$$\sum_{k\in K}\sum_{j\in V\setminus\{i\}} x_{ij}^k = y_i \quad \forall i \in V \tag{2.3.1}$$

$$\sum_{k\in K}\sum_{i\in V\setminus\{j\}} x_{ij}^k = y_j \quad \forall j \in V \tag{2.3.2}$$

These two constraints (2.3.1) and (2.3.2) say that if $y_i = 1$, for $i \in L$, then the node is entered and exited.

$$x_{0,j}^k(s_0^k + \tau_{0,j}) \leq s_j^k \qquad \forall k \in K \text{ and } j \in V \tag{2.3.3}$$

$$x_{ij}^k(s_i^k + \delta_j + \tau_{ij}) \leq s_j^k \quad \forall k \in K, i \in L \text{ and } j \in V \tag{2.3.4}$$

These constraints (2.3.3) and (2.3.4) ensures that if a bus travels between $i$ and $j$, on route $k$, then the stopping time on location $i$ is constraint to the previous time the bus has travelled.

$$s_{n+1}^k \leq M \quad \forall k \in K \tag{2.3.5}$$

Constraint (2.3.5) does not allow any route to have a travel time greater than $M$.

$$\sum_{k\in K}\sum_{j\in V} x_{ij}^k \leq 1 \quad \forall i \in L \tag{2.3.6}$$

Constraint (2.3.5) restricts more than one bus driving between $i$ and $j$.

$$\sum_{j\in V} x_{0j}^k = 1 \quad \forall k \in K \tag{2.3.7}$$

Constraint (2.3.7) ensures that a bus drives out of the factory.

$$\sum_{i\in V} x_{ih}^k - \sum_{j\in V} x_{hj}^k = 0 \quad \forall h \in L,\, k \in K \tag{2.3.8}$$

Here in equation (2.3.8) it is made sure that if a bus drives into a node it is required to drive out of it as well, if the node is in the set of locations (pickup points).

$$\sum_{i\in N} x_{i,n+1}^k = 1 \quad \forall k \in K \tag{2.3.9}$$

Constraint (2.3.9) requires all buses to end there routes at the factory.

$$x_{i,j}^k \leq \frac{\tau_{ij}}{a} \quad \forall k \in K, i \in V, j \in V \tag{2.3.10}$$

In (2.3.10) s bus must travel for a certain amount of time, $a$, before stoping at a new pickup point.

### 2.3.3 Linearity

The model, as presented in the previous section, is not linear and therefore some changes have to be made if it is to be solved in GAMS[1]. The non-linearity can be found in equations (2.3.3) and (2.3.4), where two variables are multiplied. To ensure linearity the changes listed below have to be applied, to the model.

$$s_0^k = 0 \qquad \forall k \in K \tag{2.3.11}$$
$$s_0^k + \tau_{0,j} - s_j^k = (1 - x_{0,j})W \qquad \forall k \in K j \in V \tag{2.3.12}$$
$$s_i^k + \delta + \tau_{ij} - s_j^k = (1 - x_{ij})W \qquad \forall k \in K, i \in L j \in L \tag{2.3.13}$$
$$s_i^k + \tau_{i,n+1} - s_{n+1}^k = (1 - x_{i,n+1})W \qquad \forall k \in K i \in V \tag{2.3.14}$$

Here $W$ is a large number and $W > |V|$. Other constraints are the same as in the previous section. Although when dealing with a GAMS model other constraints have to be added:

$$\sum_{i \in V} \sum_{k \in K} x_{i,0}^k = 0 \tag{2.3.15}$$
$$\sum_{j \in V} \sum_{k \in K} x_{n+1,j}^k = 0 \tag{2.3.16}$$
$$\sum_{i \in V} \sum_{k \in K} x_{ii}^k = 0 \tag{2.3.17}$$

These constraint ensure that a node does not visit itself, that no one can return to the source and that no one can leave the sink.

### 2.3.4 Upper Bounds

First and the most obvious upper bound to the problem is to let one route visit all the points.

$$UB = \sum_{i \in V} \phi_i + \beta(|K| - 1) \tag{2.3.18}$$

This upper bound requirs one bus to collect all the profits froom every node. All profits are represented by the first half of equation (2.3.18) and if only one bus is used then a profit of $\beta(|K| - 1)$ is collected from the unused buses.

#### Relaxations to travel time

The upper bound in (2.3.18) is the same for all values of $M$. Let us now incorporate $M$ into the upper bound. It is known that traveling further than $M$ from the depot is impossible. Let us now define $V_M$ as the set of all nodes closer than $M$ to the depot. The new upper bound is

$$\sum_{i \in V_M} \phi_i + \beta(|K| - 1) \tag{2.3.19}$$

[1]GAMS is a programming language used to solve linear models in operation research.

This upper bound, equation (2.3.19) does not allow profit outside the radius of maximum route length. All profit within that radius, of maximum route length, is collected with a single bus.

## 2.4 Review of Relevant Problems

The bus route problem focuses on routes that make there way through a number of pick up points before finally stopping at the last point, known as the depot. This is similar to a well known problem called the travelling salesman problem or TSP.

### 2.4.1 TSP

TSP tries to find the optimal, shortest, route from a source through a number of nodes and back to the source. A travelling salesman leaving from New York and visiting all the major cities on the east cost, of the USA, has to find the best route to travel and then return home again, hence the name travelling salesman problem.

This problem is, perhaps, the best known problem in operations research. For this problem a binary matrix is defined, $x_{ij}$.

$$x_{ij} = \begin{cases} 1 & \text{If one travels from node } i \text{ to node } j \\ 0 & \text{If one does not travel from node } i \text{ to node } j \end{cases}$$

Also a cost, $c_{ij}$, is defined. This is the cost of travelling from node $i$ to node $j$. The set of nodes is $V$ and $A$ is the set of all arcs. A model, as defined in Wosley [15] is:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.4.1}$$

$$\begin{aligned}
\text{s. t.} \quad & \textstyle\sum_{j:j \neq i} x_{ij} = 1 && \forall i \in V && (2.4.2)\\
& \textstyle\sum_{i:i \neq j} x_{ij} = 1 && \forall j \in V && (2.4.3)\\
& \textstyle\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 && \text{for } S \subset V,\ S \neq \emptyset && (2.4.4)\\
& x_{ij} \in \{0,1\} && \forall i \in V, \forall j \in V && (2.4.5)\\
& c_{ij} \geq 0 && \forall i \in V, \forall j \in V && (2.4.6)
\end{aligned}$$

Constraints (2.4.2) and (2.4.3) ensure that every node is both entered and exited. The most complicated constraint is (2.4.4) for it is a sub tour elimination constraint. The number of sub tour elimination constraints raises dramaticly with the number of nodes assigned to the problem.

The number of possible solutions for TSP, with $n$ nodes, is $(n-1)!$. Half that for the symmetric problem. The TSP problem is NP-hard [6] Wosley [15] defines NP-hard in the follwoing way: *"NP is a class of decision problems with the property that: for any instance for wich the answer is YES„ there is a "short" (polynomial) proof of the YES."* Heuristics such as Lagrangian heuristic and meta heuristics, such as tabu search, are often used to solve TSP. The largest TSP problem solved, to date, found the shortest path between 24,978 cities in Sweden [6]. To

find the solution cutting plain and branch-and-cut processes were used and it took almost a year to find the final solution [6].

As can be seen there are certain similarities between the travelling salesman problem and the bus route problem. Although in the latter it is not necessary to stop at all points but that is the case with TSP. Therefore in the bus route problem one must determine which pick up points are important and which are not. A variation of the travelling salesman problem called the price collecting travelling salesman problem, PCTSP, deals with this problem.

### 2.4.2 PCTSP

In PCTSP, each node is assigned a prize, or profit, gained when the node is visited. Not all nodes have to be visited in PCTSP but a penalty is paid for every node skipped. As in TSP $V$ is the set of all nodes.

| Name | Description |
|---|---|
| $x_{ij}$ | Is equall to 1 if the path between $i$ and $j$ is used otherwise it is 0. |
| $y_i$ | A binary number equal to 1 if node $i$ is visited else it is 0 |
| $\gamma_i$ | Penalty to be paid if node $i$ is not visited. |
| $p_i$ | Prize gained from visiting node $i$. |
| $c_{ij}$ | Cost of travelling from $i$ to $j$. |
| $B$ | A minimum amount of collected prizes. |

The PCTSP problem as presented[2] in Dell'Amico [8]:

$$\min \sum_{i \in V} \sum_{j \in V \setminus i} c_{ij} x_{ij} + \sum_{i \in V} \gamma_i (1 - y_i) \tag{2.4.7}$$

$$\begin{aligned}
\text{s. t.} \quad & \textstyle\sum_{j \in V \setminus i} x_{ij} = y_i && \forall i \in V && (2.4.8)\\
& \textstyle\sum_{i \in V \setminus j} x_{ij} = y_j && \forall j \in V && (2.4.9)\\
& y_1 = 1 && && (2.4.10)\\
& \textstyle\sum_{i \in V} p_i y_j \geq B && \forall j \in V && (2.4.11)\\
& \textstyle\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq y_h && \forall h \in V \setminus 1 \text{ and } \forall S \subset V : 1 \in S, h \in V \setminus S && (2.4.12)\\
& x_{ij} \in \{0, 1\} && \forall i \in N, \forall j \in N && (2.4.13)\\
& y_i \in \{0, 1\} && \forall i \in N && (2.4.14)\\
& && && (2.4.15)
\end{aligned}$$

Constraints (2.4.8) and (2.4.9) ensure that if a node is entered it is also exited. Constraint (2.4.10) forces the depot to be included in the cycle. In (2.4.11) a certain amount of prizes has to be gathered, a goal is defined, and (2.4.12) is a sub tour elimination constraint.

PCTSP was introduced by Balas and Martin in connection with operations of a steel rolling mill. A variant of PCTSP is the profitable tour problem, PTP. When a PTP model is

[2]Similar mathematical presentations were presented in Balas [1] and Dell'Amico [9], but a slightly different model was presented in Chaves [4]

constructed it is essentially the same as PCTSP except (2.4.10) and (2.4.11) are removed and $\gamma_i = 0$ for all $i \in V$ [9].

Many methods have been used to solve PCTSP for example Lagrangian heuristic [8] or hybrid algorithms [4].

In comparison the bus route problem and PCTSP are similar but PCTSP only allows a single route to visit pick up points. The problem presented by ALCAN can have up to five routes. A well know problem in operations research deals with multiple routes, that problem is called the vehicle routing problem or VRP.

### 2.4.3 Vehicle Routing Problem

Allocating more than one route to a number of nodes, is generally called the vehicle routing problem.

| Name | Description |
|---|---|
| $x^k_{i,j}$ | Is equal to one if route $k$ travels from $i$ to $j$ and 0 otherwise. |
| $\gamma_i$ | Penalty to be paid if node $i$ is not visited. |
| $p_i$ | Prize gained from visiting node $i$. |
| $c_{ij}$ | Cost of travelling from $i$ to $j$. |
| $B$ | A minimum amount of collected prizes. |

Routing problems are characteristically difficult to represent concisely in optimization models [12]. These problems are often very useful in the real world. A mathematical model is presented in the following way:

$$\min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x^k_{ij} \tag{2.4.16}$$

Here $K$ is the set of routes and $|K| = N$ while $V$ is the set of nodes were $|V| = n$. In this problem the depot is presented by source node, $i = 0$, and a sink node, $i = n$.

$$\sum_{i \in V} x^k_{ih} - \sum_{j \in V} x^k_{hj} = 0 \quad \forall h \in V \setminus \{0, n\},\ k \in K \tag{2.4.17}$$

$$\sum_{k \in K} x^k_{ij} = 1 \quad \forall i \in V \setminus \{0\}, \forall j \in V \setminus \{n\} \tag{2.4.18}$$

$$\sum_{i \in V} x^k_{i,n} = 1 \quad \forall k \in K \tag{2.4.19}$$

$$\sum_{j \in V} x^k_{0,j} = 1 \quad \forall k \in K \tag{2.4.20}$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \quad \forall S \subset V : 0 \in S, n \in S \tag{2.4.21}$$

$$x^k_{ij} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, k \in K \tag{2.4.22}$$

The first constraint (2.4.17) ensures that all nodes entered are exited. The second constraint (2.4.18) restricts more than one vehicle visiting any node. Then constraints (2.4.19) and (2.4.20) force all routes to leave the source and enter the sink. The subtour elimination constraint is presented in (2.4.21) and lastly (2.4.22) gives $x^k_{ij}$ a binary value.

The VRP is widely used in the real world. The best example is the delivery of goods from suppliers to customers. Here the number of vehicles and capacity of vehicles can be a factor. These problems are usually solved with a tabu search or other heuristics.

The VRP allows the use of more than one routes but the method requires each route to have the same point of origin, called a source. The routes visit all points in $V$ but must end at the same point they started from. When the routes return the source point is sometimes called a sink, this is the same location but it has two names, source and sink. In the bus route problem this is not the case, a bus can start at any node and then make its way to the depot. A variation of VRP uses the same principle. That variation of VRP is called the open vehicle routing problem or OVRP.

### 2.4.4 Open Vehicle Routing Problem

OVRP, is similar to the vehicle routing problem except when drivers have visited all nodes they do not need to return to the depot. This is similar to the bus route problem except there the bus starts at the last node and makes its way back to the depot. OVRP uses a set of Hamiltonian paths while VRP uses a Hamiltonian cycles [3]. Both a Hamiltonian cycle and a Hamiltonian path are defined in [14] as follows:

> Before defining a path or a cycle a walk must first be defined. G is a graph, a walk in G is a sequence of nodes and arcs. A path is a walk with no repeated nodes and a trail is a walk within repeated arcs. Note that all paths are trails but not all trails are paths. A circuit is a closed trail but not a path. A cycle is defined as a circuit with at least one arc and has one repeated node is $node_1 = node_n$.

In OVRP the drivers start at the depot and then finish at the last customer node. There are normally certain constraints applied to this problem. A vehicle has usually a maximum predetermined capacity and this capacity cannot be exceeded by the demand of the costumer nodes, on the route. Other constraints may also apply, for example a maximum number of vehicles or the maximum length of any single route. The OVRP has not been as extensively studied as VRP [3]. It was first mentioned, according to [3], in 1981 by Scharge in an article dedicated to the description of realistic routing problems. The mathematical formulation of OVRP is the same as for VRP except $c_{0j} = 0$, $\forall j \in V$.

OVRP is used for a number of problems, for example the school bus problem [17]. In that problem a route for school buses is determined. In [17] tabu search is used to solve the problem. Other algorithms, according to [16], that have been used include: list-based threshold accepting, BoneRoute meta heuristic and record to record travel heuristic. The last one is a deterministic variant of simulated annealing.

The OVRP has similarities with the bus route problem but it has to visit all points in $V$. In the bus route problem one is allowed to skip some nodes, pick up points, but this is not possible in the OVRP. The bus route problem does not have to visit all nodes, it does not have to begin at the source and it has to choose nodes for there importance. A problem similar to this is the team orienteering problem, or TOP.

### 2.4.5 Team Orienteering Problem

The team orienteering problem, or TOP, is a combination of PCTSP and VRP. The problem defines a set of nodes $V$, a set of arcs $A$ and a set of routes $K$, were $|V| = n$ and $|K| = N$. In this problem N routes visit n points, but does not have to stop at all points; each point has a service time and a profit.

| Name | Description |
|---|---|
| $x_{i,j}^k$ | The number of times edge $(i, j)$ transverses with vehicle $k$. |
| $y_{ik}$ | A binary number equal to 1 if node $i$ is visited by route $k$ otherwise it is 0, $i \in V$ and $k \in K$ |
| $d_{ij}$ | As the distance between two points and $(i, j) \in A$. |
| $s_i$ | Servise time ate vertex $i$, $i \in V$. |
| $p_i$ | The profit received for node $i$, $i \in V$. |
| $M$ | The total duration of each tour. |

This is in many ways similar to the bus route problem as a profit is needed for every node to determine which are to be visited. The TOP problem as presented in [13]:

$$\max \sum_{i=1}^{n-1} \sum_{k=1}^{N} p_i y_{ik} \tag{2.4.23}$$

s. t.

$$\sum_{j=1}^{n-1} \sum_{k=1}^{N} x_{0j}^k = 2N \tag{2.4.24}$$

$$\sum_{i<j} x_{ij}^k + \sum_{i>j} x_{ij}^k = 2y_{ik} \qquad \forall j \in V \setminus \{n\}, k \in K \tag{2.4.25}$$

$$\sum_{i=0}^{n-2} \sum_{j>i} d_{ij} x_{ij}^k + \sum_{i=1}^{n-1} s_i y_{ik} \leq M \qquad k \in K \tag{2.4.26}$$

$$\sum_{k=1}^{N} y_{ik} \leq 1 \qquad \forall i \in V \setminus \{0, n\} \tag{2.4.27}$$

$$\sum_{i,j \in U, i<j} x_{ij}^k \leq |U| - 1 \qquad U \subset V \setminus \{0\}, n - 2 \geq |U| \geq 2, k \in K \tag{2.4.28}$$

$$x_{ij}^k \in \{0, 1, 2\} \qquad \forall i \in V \setminus \{0, n\}, j \in V, k \in K \tag{2.4.29}$$

$$x_{0,j}^k \in \{0, 1\} \qquad \forall j \in V \setminus \{n\}, k \in K \tag{2.4.30}$$

$$y_{ik} \in \{0, 1\} \qquad \forall i \in N \tag{2.4.31}$$

$$\tag{2.4.32}$$

The first constraint (2.4.24) ensures that $N$ tours leave the source node and then return. To ensure connection of selected nodes is (2.4.25) and (2.4.26) limits the length of any single tour. The constraint (2.4.27) prevents more than one route going through a single node, other than the depot. The sub-tour elimination constraint is (2.4.28). The last three constraint show allowed values for the variables.

The TOP lets one construct $N$ routes through $n - 1$ nodes and the depot. Stopping at any single point gives a penalty or service time. This could for example be used for routing technicians to service customers at geographically distributed positions.

The TOP is a NP-hard problem as it is a varition of the selective traveling salesman problem [13]. Methods used to solve TOP include tabu search [13], greedy construction procedure

and 5-step heuristic. A single route TOP, called the orienteering problem or selective traveling salesman problem, has been solved with up to 500 nodes. This was done using branch-and-bound and branch-and-cut [13]. Some times TOP is referred to as multiple tour maximum collection problem. Of all the different problems presented the TOP is most similar to the bus route problem.

## 2.5 Review of Methods

### 2.5.1 A Lagrangian heuristic for the Prize Collecting Travelling Salesman Problem [8]

An article inspecting how to solve PCTSP with a Lagrangian heuristic by M. Dell'Amico, F. Maffioli and A. Sciomachen. A good introduction to the PCTSP. The underlying cunstruction of the bus route problem, presented in the report, is based in partially on the PCTSP model in this article. The problem presented in the article is minimized. Therefore to assist in determining the valitity of calculated solutions a lower bound was also calculated. This lower bound is found in [9]. A feasible solution is found by using Adding-Nodes Procedure where two rules, R1 and R2, are compared. From these comparisons R2 was shown to be better in this instance. This feasible solution is then defined as an upper bound as no feasible soultion with a lower value objective value is know.

To improve upon feasible solutions two methods are combined. The first was the so called Extension phase tries to improve the overall profit of the current cycle. The second method was called Collapse phase and it tries to remove the most expensive node each time. Together the method was called Extension and Collapse. Lastly a Lagragian heuristic was developed so that Extension and Collapse was applied in each computation of the Lagrangian multiplier. This method was then used on a few computational experiments. The conclusion of the experiments was that with increased profit, that needs to be collected, the computational time required increased while the quality of the solutions decreases. This quality of solutions was mesured as the ratio between upper bound and lower bound.

### 2.5.2 Price Collecting Travelling Salesman Problem [1]

This is an article by E. Balas concerning the price collecting travelling salesman problem. It was Balas who, along with Martin, first introduced the PCTSP. There is an introduction to PCTSP in its first section. After this the article becomes very mathematical and complicated.

The main focus of this article is to discuss the structural properties of the PCTSP polytope, the convex hull of the solutions to the PCTSP.

### 2.5.3 On Prize-Collecting Tours and The Asymmetric Travelling Salesman Problem [9]

An article by M. Dell'Amico, F. Maffioli and P. Värbrand. The article contains a short introduction to PCTSP and a model is presented. There is also a definition for PTP, profitable tour problem; and APTP, asymmetric profitable tour problem. This article featured a good

section on tests which proved to be helpful in conducting tests for the model inspected in this report. Test were randomly generated.

The article defines PTP by removing certain constraints from PCTSP and allowing the empty solution. A simple heuristic is defined to solve PTP. It is also discussed how the PTP can be polynomialy reduced to Asymmetric TSP on a large diagraph. Three previously discovered lower bounds for PCTSP are presented and also a new lower bound for PCTSP is put forth. For asymmetric PTP two lower bounds are presented by removing constraints. The article ends with a section on computational experiments both for PTP and PCTSP. Were all instances were solved in less than one minute of CPU time. It was also concluded, by inspecting ratios between lower bounds, that solutions to large asymmetric PTP problems were good.

### 2.5.4 Hybrid algorithms with detection of promising areas for the prize collecting travelling salesman problem [4]

This article by Agusto and Lorena on PCTSP presents some ideas of clustering, using evolutionary cluster search and a hybrid approach called CS*. This hybrid approach was constructed from Greedy Randomized Adaptive Search Procedure, or GRASP, and Variable neighbourhood search. The methods are given a short description and how they can solve PCTSP is explained. These ideas could be useful in further development of insert moves or bus moves. The article starts with an introduction where PCTSP is introduced and a short history of the problem is given. The next section puts forth a mathematical model of PCTSP, this model is a little different from the one in [8]. In the third section ECS, evolutionary cluster search, and its components, evolutionary algorithm, interactive clustering, analyzer module and a local search; are explained. Then a section describes how ECS is applied for PCTSP. The hybrid approach called CS* is then applied to PCTSP. In this section a few interesting moves are defined. These 6 moves were different from the ones used in this project. One move called $m_4$, is comparable with *insert move* 13[3]. Other moves were similar but often used more nodes, for example $m_1$ inserted 2 nodes instead of one. The last section is on computational results and show solution from ECS and CS*. The results from these two are also compared to results from a CPLEX 7.5 solver. In conclusion the authors find that CS* returns better solutions and use of these methods is validated.

### 2.5.5 A tabu search algorithm for the open vehicle routing problem [3]

This article by Brandao contains a good introduction to OVRP and compares it to VRP. Most of the information in the section on OVRP came from this source. There is also a short introduction on the history of OVRP and relatively few, compared to VRP, have studied it. The meta-heuristic used in the article is tabu search. The importance of a good initial solution is discussed and how to attain such a solution, the methods used for this are nearest neighbour heuristic, or NNH, and a solution based on a pseudo lower bound. The pseudo lower bound is a method based on minimum cost spanning tree with degree k subject to relaxations. Initial solutions given with an insertion heuristic and a lower bound were experimented upon. Before applying the tabu search to this initial solution the solution is submitted to one of two methods: nearest neighbour or unstringing and stringing method. This was done to improve the solution. In the tabu search swap and insert moves are used. The goal of the algorithm was

[3]The moves and neighborhoods are defined in the next section.

to minimize the number of routes and therefore new routes could not be created. A method was included that tried to join the two routes with the lowest demand. This is clever and could be implemented to the algorithm used in the report in the future. In conclusion it is stated that the algorithm gave good solutions for a very short computing time, outperforming former algorithms such as the one proposed by Sariklis and Powell. For example the method of using psuedo lower bound gave an average travel time of 416.1 while Sariklis and Powell algorithm had an average travel time of 488.2. These are from calculations with 50 point data sets and the differance in running times was 88.6 seconds, Sariklis and Powell method solved the problem in 0.22 seconds.

### 2.5.6 Open Vehicle Routing Problem with Time Deadlines: Solutions Methods and Application [17]

This article, by Aksen, Aras and Özyurt; focused on the OVRP with time deadlines, or OVRP-TD. Clarke-Wright parallel saving algorithm modified for OVRP was implemented along with greedy nearest neighbour algorithm and a tabu search heuristic. The article also contains a short description for most of these methods. The article explained how Clark-Wright, CW, is modified for OVRP-TD, mostly by setting certain distances to infinity. Then CW and the nearest neighbour algorithm were used to find an initial solution. There neighbourhood consisted of three moves, which were 1-0 move, 1-1 exchange and 2-Opt move. These three moves are the same as the swap moves described in this report. Local search with these moves is incorporated into TS as a tool of local post optimization, LPO. The chapter on computational results solving five random results and one real problem , a school bus problem in Istanbul. In conclusion it was apparent that CW initial solution performed better than classical heuristics with LPO. Overall this is a very short article that does not go much into details.

### 2.5.7 A general heuristic for vehicle routing problems [11]

This article, by Pisinger and Ropke, is a large, extensive and takes on various vehicle routing problems. VRP with time windows, capacitated VRP, multi-depot VRP, site dependant VRP and OVRP are all discussed and solved by transforming each instance into a single type of model. The model is called Rich Pick up and Delivery Problem with Time windows, or RPDPTW. There is a mathematical presentation of this model that is a little confusing, on account of the number of sets involved. All the models RPDPTW solves are VRP models and therfore have to visit all nodes presented in the system, which means the RPDPTW can not be appplied to the bus route problem. Next there is a section on how one transforms these five different VRP problems into a RPDPTW. This article and the model presented are good reading material when presented with a problem as discussed in this report. The article also explains different objectives of its model. The first objective is to minimize the number of vehicles while the second objective is to minimize the travel distance. This is in accordance with the problem presented in this report where the first objective is to visit as many nodes as possible, with given travel constraints, while using as few buses as possible and the second objective is to minimize the travel distance/time. The heuristic used to solve RPDPTW is adaptive large neighbourhood search, ALNS, a method that uses two, a constructive and a destructive, neighbourhoods to find an optimal solution. It is explained how one applies the ALNS to RPDPTW and then there is a large section on computational results. In conclusion

it is stated that the ALNS should be considered as one of the standard frameworks for solving large-sized optimization problems, as the method is very general and gave good results.

### 2.5.8 Open vehicle routing problem with driver nodes and time deadlines [16]

This article looks at a particular variant of the OVRP where the vehicles, routes, start at the depot and visit a number of nodes but all routes are required to end at certain types of nodes called driver nodes, this problem also has time deadlines that have to be kept. A mathematical model is presented for this particular type of problem. The problem is quite different from the one presented in this report but as with articles on similar subjects it is worth a look to get a better understanding on OVRP.
The introduction section in this article, by Aksen, Aras and Özyurt, contains an excellent historical overview of OVRP. Instrumental articles and methods used are mentioned. The authors also state that they know of no other article where a similar problem, OVRP using driver nodes, is tackled. To solve the problem a new heuristic called open tabu search is used. It makes use of three move operators in generating the solutions in the neighbourhood of the current solution. These moves are the same as defined in [17]. The initial solution is found with a nearest insertion heuristic and a Clark-Wright parallel saving algorithm. The problem called OVRP-d is mathematically presented as a mixed integer problem in the second section. This is clearly presented and not complicated. The next section is on the tabu search algorithm previously described. The forth section is on computational results where the open tabu search, OTS, is compared to various classical heuristics. Then in conclusion it is determined that the new heuristic, OTS, gives higher quality solutions then the classical heuristics.

### 2.5.9 A TABU Search Heuristic for the Team Orienteering Problem [13]

This article, by Tand and Miller-Hooks, on the team orienteering problem was very useful for the project. The team orienteering problem, TOP, is very similar to the model presented in this report. Also the authors supplied data so comparison tests, between their results and the algorithm in this project, could be performed.

The article starts out with a good introduction to TOP. The connection between TOP and several other problems is discussed. Also the method that have be inspected when solving TOP are listed, simulated annealing is not one of them. The next section puts forth the mathematical model in a very straight forward manner. The article explains how the initial solution is calculated with a method known as adaptive memory procedure, AMP. This is an excellent method for calculating an initial solution, although might in some cases be problematic if the best solution is using no routes[4]. Interestingly the tabu search algorithm uses intermediate infeasible solutions to aid in the search process, by moving solutions out of local optimums. Other methods like small and large neighbourhood search and methods used for tour improvement are also discussed. The section on computational results shows comparison between TABU search, 5-step heuristic and a version of the Tsiligirides heuristic extended for TOP by Chao. In conclusion it is noted that AMP and its mechanism, alternating between small and large neighbourhoods stages and using both random insertion and greedy procedures led to an effective tabu search algorithm.

[4]For example in the algorithm used in this project.

# Chapter 3

# Simulated Annealing for the BRP

## 3.1 Simulated Annealing Algorithm

Now the model for the bus route problem has been put forth, similar problems explored and solution methods for those methods discussed. The other problems, that were compared to BRP are solved with a heuristic or a meta-heuristic methods. Also in [13] it is shown that TOP is NP-hard as it is a special case of the selective travelling salesman problem[1]. Now if, in the bus route problem, $\beta = 0$ the we have the TOP problem with $\tau_{0,j} = 0$, $\forall j \in V$. Therfore the bus route problem is NP hard as well. In light of this it is necessary to choose a heuristic or meta heuristic to solve the problem. In computer science one strives to find as good a soltution as possible using as short a running time as possible. A heuristic is an algorithm that sacrifices one or both of these goals.

Meta-heuristic is a method used for solving a class of computational problems, which are common in operations research. To find the best method suited for solving the BRP a simple but effective meta-heuristic was needed. Simulated annealing is one such method and it has given good results in the past, when dealing with similar projects. Therefore simulated annealing algorithm was used to solve the BRP.

The idea of simulated annealing is to look at different solutions and compare them and accept the better solution, except in certain instances a worse solution may be accepted. A pseudo code of simulated annealing is given in [15] and it is presented below:

[1] Selective TSP is a variation of PCTSP.

*Pseudo Code*

1. Get an initial solution S.
2. Get an initial temperature, $T_0$, and a reduction factor, $r$, with $0 < r < 1$.
3. While not yet frozen do the following:
   (a) Perform the following loop L times.
      i. Pick a random neighbor $S'$ of $S$.
      ii. Let $\Delta = f(S') - f(S)$.
      iii. If $\Delta \geq$ set $S' = S$.
      iv. If $\Delta < 0$ set $S' = S$ with probability $e^{-\Delta/T}$.
   (b) Set $T \leftarrow rT$. Reduce temperature.
4. Return the best solution found.

The value of $T$ is used to calculated $e^{-\Delta/T}$, this is the probability that determines if a worse solution will be accepted or not. The reduction factor, $r$, determines how fast the values of $T$ will drop in each iteration. This along with a frozen value and stopping criteria is called a cooling schedule. There are many different types of cooling schedules and some of them are discussed in [10]. The basic idea of the cooling schedule is to minimize the likelihood of the optimal value being a local optima and not a global optima.

The cooling schedule chosen in this project is the one described in Wosley [15]. Most cooling scheduls would be effective for this problem, therefore the chosen schedule is just as good.

### 3.1.1 Neighborhoods

In the description of the simulated annealing algortim a solution $S'$ is defined as a neighbor of $S$. This means that a similar solution to $S$, containing almost all the same nodes as $S$. This similar solution $S'$ is there fore defined in the neighborhood of $S$. In problems such as the BRP there are a few common neighborhoods. These are:

1. **The insert move:** One, or more, nodes are added to a possible solution. That is if the algorithm chooses to add $node_h$, $h \in V$, then $node_h$ is in $S'$ but not in $S$. Therefore $node_h$ has been inserted into the possible solution. In this report this move is reffered to as insert move 11,12,14 and 15.
   (a) **insert move 11:** Randomly selects an unused node into the solution.
   (b) **insert move 12:** Selects the highest profit unused node with the lowest possible node number[2].
   (c) **insert move 14:** Selects the highest profit node farthest from the depot.
   (d) **insert move 15:** Selects the highest profit node closest to the depot.

[2]Nodes have number ranging from 0 to $|V|$.

2. **1-1 move:** Two selected nodes, that are in $S$, are swaped by preserving there original positions. In this report this move is reffered to as swap move 11.

3. **Insert and Remove**: One selected node is removed from the route and another unused node is inserted into the route. In this report this move is referred to as insert move 13.

Other possible neighborhoods are:

1. **Swap move 21**: Here two selected nodes, in two separate routes, are randomly chosen and exchanged, preserving their original possition.

2. **Swap move 31**: Here a selected node is removed from a route and inserted randomly into another separated route.

3. **Bus move**: If there exists a route containing no nodes then a random number of unused nodes will be selected and inserted into that route.

These six moves form the neighborhood used in the simulated annealing algorithm. The first three were chosen as they are often used in the literature. The different types of *insert moves* were devised as it is one of the, if not the most, important moves. This is manly because the initial guess is the empty solution, no active routes, and therfore the algorithm has to construct the routes. *Insert moves* add new nodes to routes thereby increasing there profit and the value of the objective function.

When defining *insert move* 14 one could have defined two parameters determining what is high profit and what is far from the depot. Instead a more linear approach was chosen, simply as it was more straight forward and easier to program. This linear approch defined the profit of a node as $\phi_i$ and the distance between $node_i$ and the depot was defined as $\tau_{0,i}$. To determine which node to choose *insert move* 14 inspected $\phi_i \tau_{0,i}$.

Similarly two parameters could have been defined for *insert move 15*, determining what is high profit and what is close to the depot. A linear approach was also used in this case as it was logical and easy to program. The values inspected by *insert move 15*, to determin what node to choose, was the ratio $\phi_i / \tau_{0,i}$.

In *insert move 13* the lowest profit node was removed from a randomly selected route. Then another node, chose randomly, was inserted into the selected route. It might be wiser to remove a random node rather than one with low importance, but this was not implemented due to time constraints.

The *bus move* may be a bit crude but it was devised to speed things along in the first iterations of simulated annealing. By using the *bus move* entire routes were added and thereby decreasing the number of iterations needed to construct them simply by using *insert move*.

The *swap move* 31 was deemed necessary. None of the other *moves* could assist in the removal of a route so this one was constructed. Other possibilities were inspected, for example removing a whole route and distributing its nodes to the remaining routes. This was not considered optimal and could potentially do more harm than good.

Other *swap moves* were also used, called *swap move* 11 and 12. They moved nodes around in the routes thereby attempting to decrease travel time.

### 3.1.2 Adapting SA for BRP

In simulated annealing one inspects the objective value, always accepting a better solution and with a probabilty of $e^{-\Delta/T}$ accepting a worse solution. In the BRP the objective function is not the only thing inspected. Let us define $\omega_i$ as the combined route lengths of all routes in a solution at iteration $i$. Then if $\Delta = 0$ there is a chance that $\omega_{new} < \omega_{old}$. This new solution $S'$, with travel time $\omega_{new}$, may not return a higher objective value but is none the less a better solution. Therefore small changes were made to the simulated annealing algorithm. This update was introduced late in the project and therefore not implemented in all tests, before this was programmed the algorithm used a simulated annealing algorithm with only one $\Delta$. This updated version of the algorithm is from here on called *the updated simulated annealing algorithm.*

*Pseudo Code*

1. Get an initial solution S.
2. Get an initial temperature, $T_0$, and a reduction factor, $r$, with $0 < r < 1$ .
3. While not yet frozen, maximum number of iterations is not reached, do the following:
   (a) Perform the following loop L times.
      i. Pick a random neighbor $S'$ of $S$. Where one of the neighborhoods is chosen.
      ii. Let $\Delta = f(S') - f(S)$.
      iii. If time constraints are not broken then do the following:
         A. If $\Delta > 0$ set $S' = S$.
         B. If $\Delta = 0$ and $\omega old > \omega new$ set $S' = S$.
         C. If $\Delta = 0$ and $\omega old \leq \omega new$ set $S' = S$ with probability $e^{-\Delta_2/T}$.
         D. If $\Delta < 0$ set $S' = S$ with probability $e^{-\Delta/T}$.
   (b) Set $T \leftarrow rT$. Reduce temperature.
4. Return a solution.

The cooling schedule used in this pseudo code is the one used in Wosley [15]. The initial values of parametere, of the cooling schedule, were set to $T_0 = 3000$, $r = 0.5$ and the stopping criteria $F = 0$. The author had little experience with this to begin with that led to this bad choice. In Wosley [15] it says that the reduction factor is a positive number less than one, $0 < r < 1$. Therfore an average of $r = 0.5$ was chosen.

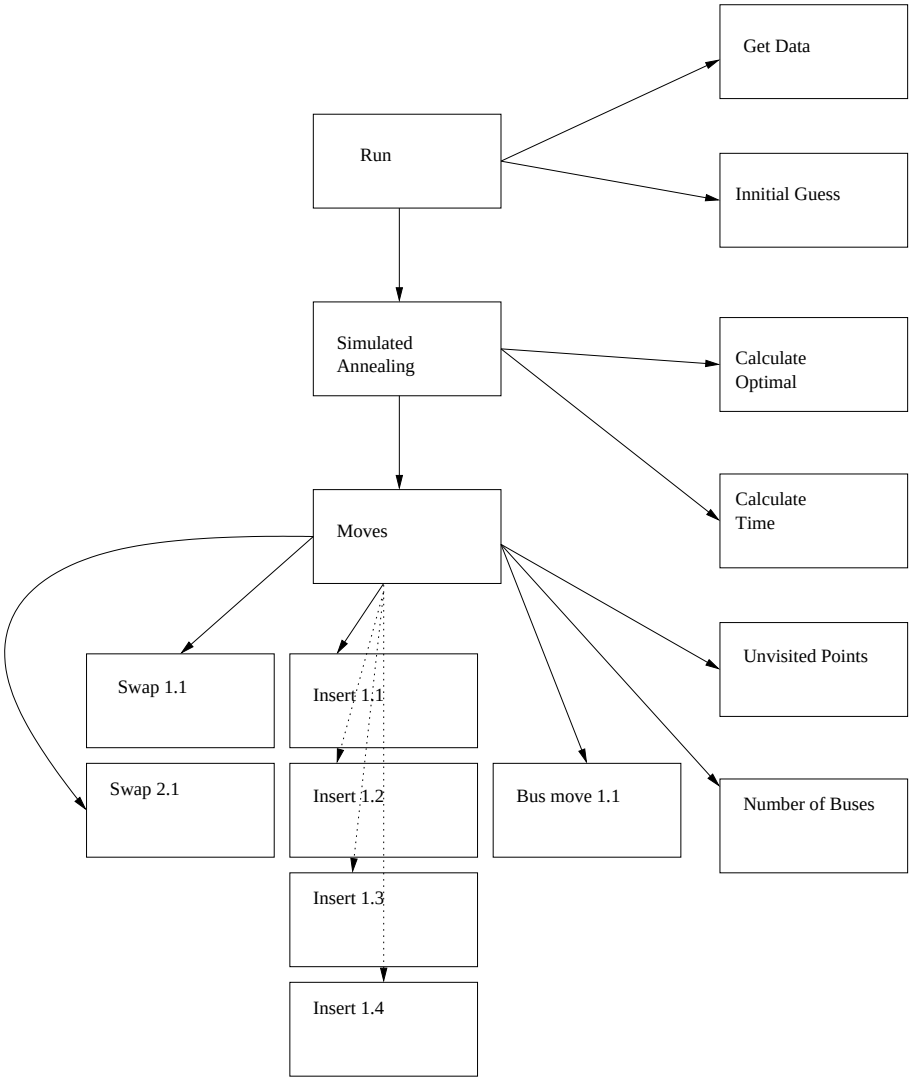



Figure 3.1: Shows how classes call other classes.

## 3.2 Implementation Details

Programming was done in the Java programming language, as the author had previous experience with the language. It was decided that using CPLEX[3], applicable with C and C++, would not be an option as ALCAN did not have the program and a commercial license is expensive.

The algorithm is divided into a number of classes. Of the classes there are two most important: SimulatedAnnealing.java and moves.java. These two classes are the core of the algorithm. One can see how the classes are called in Figure 3.2.

[3]CPLEX is an optimization software package.

## 3.3 Classes

All the classes have different roles in the whole algorithm, they can all be seen in appendix $C.2$.

### 3.3.1 Run.java

This class is the main file, it is used when the algorithm is to be run. In this class information defined from input files, using GetDataFrom*.java; the initial guess is defined, in Initialguess.java; and simulated annealing is performed, in SimulatedAnnealing.java. The whole run of the algorithm is timed to see how long a calculation takes. Many of the constants used in the program are defined in Run.java and therefore it is crucial to change the file if a different data set is being tested.

The initial guess generated in this project was an empty set, all buses driving from source to sink. There are other possibilities for generating this guess, for example in [13] a method called adaptive memory procedure is used to find initial guesses.

### 3.3.2 SimulatedAnnealing.java

This class is the core of the algorithm as it performs the simulated annealing. This class calls moves.java, CalculateOpt.java and CalulatedTime.java. All of the time constraint are handled in this class. The temperature, reduction factor, frozen factor, maximum number of iterations and the maximum travel time are defined in this class.

### 3.3.3 Moves.java

This class calls the neighborhood classes, UnvisitedPoints.java and NumberOfBuses.java. In this class the probabilities of certain neighborhoods are determined. This is done by using a probability matrix, $P$.

### 3.3.4 Neighborhood Classes

There are six previously defined neighborhoods.

In BusMove.java a *bus move* is implemented. If the new proposed solution is infeasible, for example the route too long, SimulatedAnnealing.java will reject it.

The three types of swap moves are called in SwapMove11.java, SwapMove21.java and SwapMove31.java. As with the *bus move* if any of the solutions calculated by the *swap moves* are infeasible SimulatedAnnealing.java will reject it.

There are four different *insert moves* (11, 12, 14 and 15). These are defined in InsertMove11.java, InsertMove12.java, InsertMove14.java and InsertMove15.java. As with the othe moves if a solution is infeasible the soltuion is rejected. Which *insert move* is best suited for the algorithm is determined in tests.

InsertMove13.java was created for a certain case. If an unimportant node was added to the solution there would be a chance that this node would be removed and replaced with a more important node.

### 3.3.5 Other Classes

Two classes called calculateTime.java and calculteOpt.java calculate the travel time of the routes and the objective function. In calculatedOpt.java both constants in the objective function, $\alpha$ and $\beta$, are defined.

NumberOfBuses.java is used to determine how many routes are currently active. This has to be used for example in BusMove.java, because adding an already active route is impossible.

UnvisitedPoints.java is definitely the bottle neck of the program. It uses a triple for loop to construct a vector of unused points. This is necessary in the program, for one cannot add a point that is currently in use. The class uses the two dimentional matrix route to determine which points are not in use. Route is a $|K| \times |V|$ matrix that shows the all the routes and the points they visit. In an earlier version of UnvisitedPoints.java a vector called $Y$ was used. This version was simpler and faster but unfortunatetly because of inheritance factors in Java this did not work. Future inspections of the program might fix the problem but in this project too much time had been spent on the problem so it was left as is.
A second version of UnvisitedPoints.java was constructed that removed all points within a certain radius of a chosen node from the set of usable nodes.

Very late in the project's process an important class was created called Decrease.java. The objective of this class was to inspect which points where within radius $M$ from the depot and remove all other points. As it is impossible for a route to travel further than $M$, maximum route length, because all points at a further distance are unimportant. This class was a great sucsess decreasing runtime from over 100 seconds to under 10 seconds in one instance[4]. This class also provided better solutions. Unfortunatetly the class was introduced late in the project so so it was not included in all tests, those test that used this class indicate so in there introduction.

Programs were also constructed, during experiments, to run a number of tests consecutively. These were all simple programs only constructed for optimal use of time.

[4]Data set 50a, $M = 20$

# Chapter 4

# Tests

## 4.1 Data Sets

There were 25 data sets and of those 22 were constructed. The other three were obtained data sets from Tang and Miller-Hooks [13].

To verify the algorithm, used for solving the bus route problem, tests were implemented. Note in this chapter $\tau_{ij}$ are always Euclidian distances, also note that $\tau_{0,i} = 0, foralli \in V$ when dealing with generated data sets. Also $\delta_i$, penalty for stopping at a single node, can have differrent values for all $i \in L$. In the test performed for constructed data sets $\delta_i = 1, \forall i \in L$. In test using obtaind data sets $\delta_i = 0$ for all $i \in L$. For the depot $\delta_0 = 0$ and $\delta_{n+1} = 0$ in all tests.

### 4.1.1 Constructed Data sets

The constructed data sets are catagorized in to two types, the non-randomly constructed and data sets and the randomly constructed data sets.

The non-randomly constructed data sets were situated in a graph of the scale $100 \times 100$. The depot was defined as the center, $(50, 50)$, and had a number of routes in a certain direction. The possible number of these routes was 3 and 4. A typical data set with three routes can be seen in Figure 4.1. Each node was situated so that it had the coordinates $(m_x, m_y)$, where $m_x \in \mathbb{Z}_+$ and $m_y \in \mathbb{Z}_+$ (both positive integer numbers). This means that when using three routes, originating from the depot, the Euclidian distance between a point and its closest neighbor was either 1 or $\sqrt{1+1}$. In the case were there were four routes, originating from the depot, the distance between a point and ist closest menighbor was always 1.

Of the non-randomly generated data sets six had 50 points and six sets had 100. As these data sets where supposed to be simple, to inspect if the algorithm works for the simple cases, the profit of every node was the same, with a value of 10. This was perhaps a bit to simple so a number of nodes had to be removed or have there profits decreased, otherwise the data sets would ahve been to simple. Therefore a small number, 10%, of points were chosen. the reason for choosing 10% is that 10% is sufficently small to change the data set but still retain the original simplicity. This means that in a 100 points data set 10 were selected. Of these ten points five were removed and five had there profits decreased to 1. When dealing with

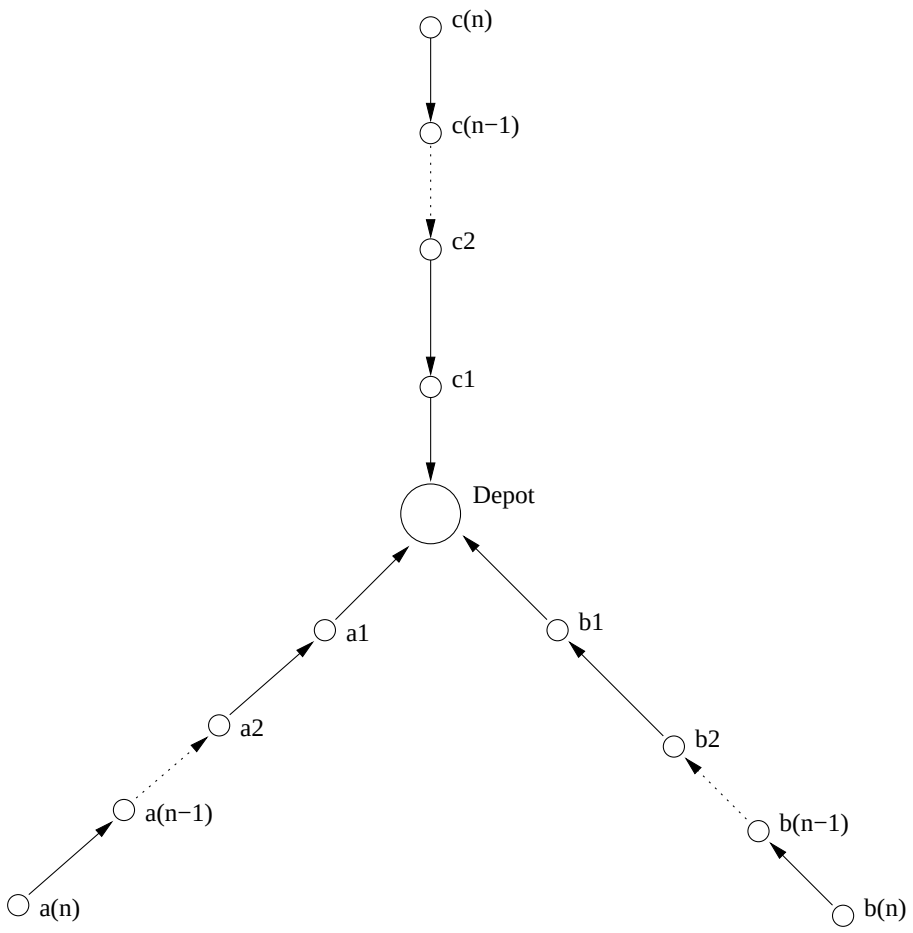



Figure 4.1: This is a test, type 1, with three routes and $n$ points on each route.

50 point data sets five points were selected, of these three were removed and two had there profits decreased to 1.

The non-randomly generated data sets were named by three parameters. First was the number of routes, second was the number of points and third was the data sets number among similar data sets. Therefore a three route 50 point data set generated second, among other 50 point three route data sets, was named: 3_50_$b$. There were only three similar data sets a,b and c.

The randomly generated data sets were devided into two subgroups, the 50 point data sets and the 100 point data sets. The 50 point data sets were generated on a $50 \times 50$ graph, as with the non-random data sets each coordinate consists of two positive integere numbers. In the data sets the depot was defined in one corner, $(50, 50)$. This is similar to the location of ALCAN aluminium plant with concern to Reykjavík. Of these 50 points each was given a profit ranging from one to ten, One representing an node of unimportance and ten a node of great importance. This scale was used because it has, in the past, been used in similar situations with good result. These profits were randomly generated in Java.

The 100 point randomly generated data sets were situated on a $100 \times 100$ graph. The depot for the 100 point data sets is defined as the point $(50, 50)$. As with other data sets the coordinates consisted of two positive integer numbers. Profits ranged from one to ten.

The randomly generated data sets were named according to there rank among similar data sets and the number of points in them. The ranks were defined as a,b,c,d and e where $a$ was genrated first and $e$ last. Therefore a 100 point data set genrated fourth was called data set 50$c$.

A subset of data set 50a was also constructed. This subset conssited of the 20 first points in the data set 50a. The set was named data set 20.

### 4.1.2 Obtained Data Sets

When constructing computational experiments it is necessary to compare your results to other similar methods. In this project the TOP was most similar to the BRP and thus ideal for comparison. The authors of [13] were kind enough to send supply data sets so that a comparison could be done. The three data sets used had 102 points, 32 points and 33 points. All of these were randomly generated and had profits ranging from five to 50[1].

## 4.2 Cooling Schedule

In Simulated Annealing a cooling schedule with three parameters, must be implemented. These three parameters are the temperature, $T$; reduction factor, $r$; and the definition of frozen[2], $F$. To find the best combination of these three parameters they must be tested. In one test $T_0 = 3000$ and $r = 0.5$ along with the forzen value of $F = 0$ was used. This cooling schedule is very fast as can be seen in Figure 4.2, so fast in fact it has nearly the same qualities as no cooling schedule what so ever. Fortunatly it was only used in one test and unfortunatly due

[1] The was no reason given in [13] for the choice of these profits.
[2] A stopping criteria.

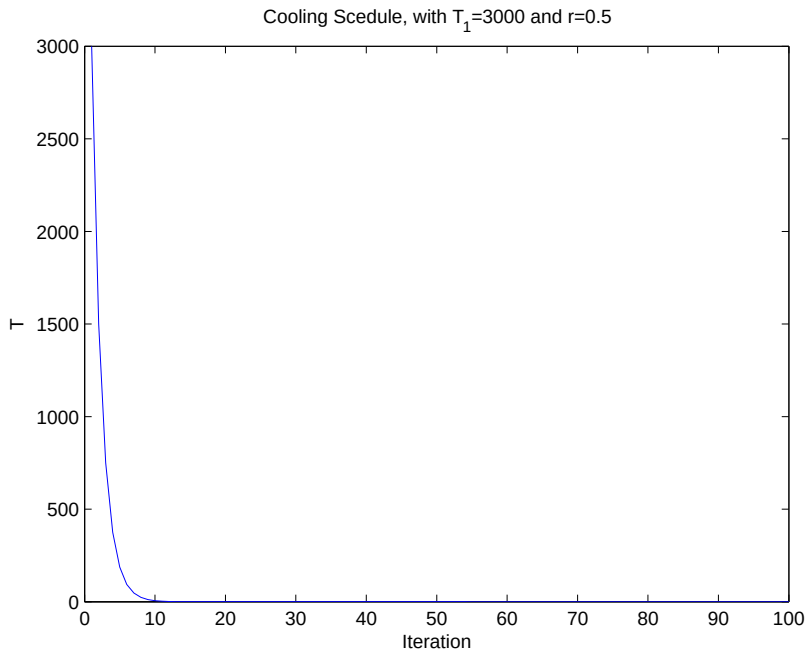



Figure 4.2: Shows the cooling schedule with $T_0 = 3000$ and $r = 0.5$, note that $T_{100} = 4.7332 \cdot 10^{-27}$.

to time constrants that test could not be repeated.

Now the initial solution to this problem is $f(S) = |K|\beta$ and if the first iteration is a *insert move*, that chooses a node with profit $\phi^*$, then the new solution is $f(S') = \phi^* + (|K| - 1)\beta$. Now $\Delta = f(S') - f(S) = \phi^* - \beta$. It is know that for all nodes in $L$, constructed data sets, the profit $p_i \in \{1, 2, 3, ..., 10\}$. If values $\Delta$ are inspected, see Figure 4.3, $\phi_i = 5$, for $i \in L$, is the most common value for profit, therfore $\Delta = -10$ is the most common value for $\Delta$. This is only because $\phi_i = 5$ is the most common value.
Somtimes, when a *bus move* is implemented, $-5 \leq \Delta \leq 0$. The reason for this is that *bus move* inserts more the one node into the solution.

An initial test with data set 50a determined a cooling shecule. That schedule was then retuned with data set 50a and then again with data set $3\_50\_a$.
In most test a value called residual ratio was inspected. This is now defined. Let us say that $Obj_i$ is the result given by a single run and that the best known value is $z$. Then the residuals are defined as:

$$r_i = z - Obj_i \quad \forall i \in T$$

Here T is the set of trials, for each individual $M$. So the residual ratio for each $M$ is defined as:

$$\frac{\sum_{i \in T}(z - Obj_i)}{|T|} \tag{4.2.1}$$

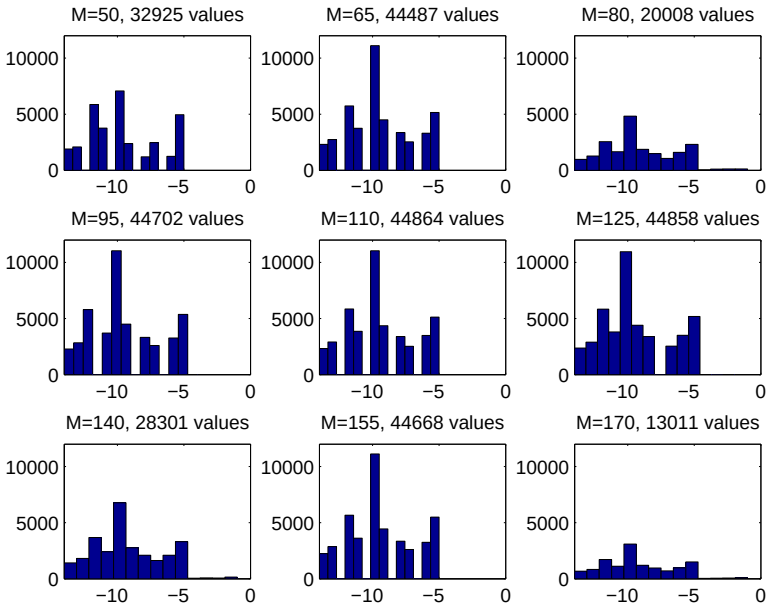



Figure 4.3: Histograms for 50 points and 4 routes. Shows how many negative values, vertical axis, of $\Delta$, horizontal axis, were calculated.

Where $|T|$ is the number of trials for each individual $M$.

### 4.2.1 First Trials for Calculating a Cooling Schedule using Data Set 50a

To find the best $T_0$ a set of possible $T_0$ was constructed and tested on data set 50a. To begin with the reduction factor was set to $1 - 10^{-9}$ and the stopping criteria set to 1, so when the temperature goes below 1 the algorithm will start using strictly local search. The values of $T_0$ tested were 28 and ranged from 6 to 10,000.

These initial values for $r$ and $F$ were found in a discared test using data set 50a. There 20 trials were used for each $M$.
After the finding a good initial temperature a good reduction factor was calculated. The new better $T_0 = T_0^*$ was used and the stopping criteria was still kept at 1. There were 14 different values of $r$ tested ranging from 0.99 to $1 - 10^{-14}$ along with $r = 0.5$.

To find good stopping criteria a test was performed inspecting different values of the factor called frozen[3]. In this inspection the better values of $T_0 = T_0^*$ and $r = r^*$ were used. There were 17 values inspected ranging from 10 to $10^{-10}$.

Each value was tested for 9 values of $M$, maximum route length, in these trials and each test

[3]The stopping criteria.

was performed 50 times. The maximum number of routes was 3. In these test *the updated simulated annealing* was used.

### Results for Initial Temperature, $T_0$

To find the best initial temperature residual ratio was inspected. The reason is so that some referance of the performance could be established. This was done by comparing all results found from the data set 50a while $M \in \{10, 20, 40, 50, 70, 80, 100, 130, 160\}$. At this time the best known objective for these values of $M$ and using three routes are shown in Table 4.2.

Table 4.1: Best known values for the data set 50a

| **M** | 10 | 20 | 40 | 50 | 70 | 80 | 100 | 130 | 160 |
|---|---|---|---|---|---|---|---|---|---|
| **OPT** | 45 | 49 | 102 | 128 | 172 | 194 | 241 | 258 | 258 |

Table 4.2: Shows the best known objectives for data set 50a. For $M = 130$ and $M = 160$the objective value is 258, which is the combined profit of all nodes in the data set.

Now to compare different temperatures plots were made of each. In these plot of the best known objective value was compared to the average objective value obtained, with that initial temperature, and the residual ratio was also inspected. These figures can be viewed in appendix $D$.2.1. As always the results with the result which gave the smallest residual ratio was the one considered best. Four different temperatures and their results can be viewed in Figure 4.4.

When these four graphs are compared it may be hard to see which one gives the best results. So all residual ratio values were summed and the result giving the lowest value, for the sum, is the best result. This, the sum with the lowest value, was found when $T_0 = 15$. When all temperatures were compared the solution was by far the lowest. The second lowest sum, of residual ratios, was found when $T_0 = 19$. If residual ratios, for all values of $M$, are summed up then one can inspect how they change in connection with temperature in Figure 4.5. The value for $T_0 = 15$ is considerably lower than the other values and outside of the 10% interval. The $\pm 10\%$ intervale, from the mean, is to show more clearly which $T_0$ stand out. Other plots that show the same graph with the mean of all residual sums and a plot of all the residual sums can be seen in the appendix.
For further trials, inspectinge the cooling schedule, $T_0 = 15$ is used unless otherwise specified.

### Results for Reduction Factor, $r$

All results from the test be seen in appendix $D$.2.2. Figure 4.6 shows the four best results, these four have the lowest average value for there residual ratios. Of the four, and all other inspected $r$ values, the value $r = 1 - 10^{-13}$ gives the lowest values of residual ratio.

Let us now inspect Figure 4.7. This shows the total sum of the residual ratios for each instance and all $M$. The instances are $\{0.5, 1 - 10^{-2}, 1 - 10^{-3}, 1 - 10^{-4}, ...1 - 10^{-13}, 1 - 10^{-14}\}$. When this plot is compared to the one in Figure 4.5 one can see that they are not the same when $T_0 = 15$ and $r = 1 - 10^{-9}$, that is instance 7 in figure 4.7. This is due to the fact that there are a lot of random factors in the algorithm. In Figure 4.7 the first instance of reduction factor

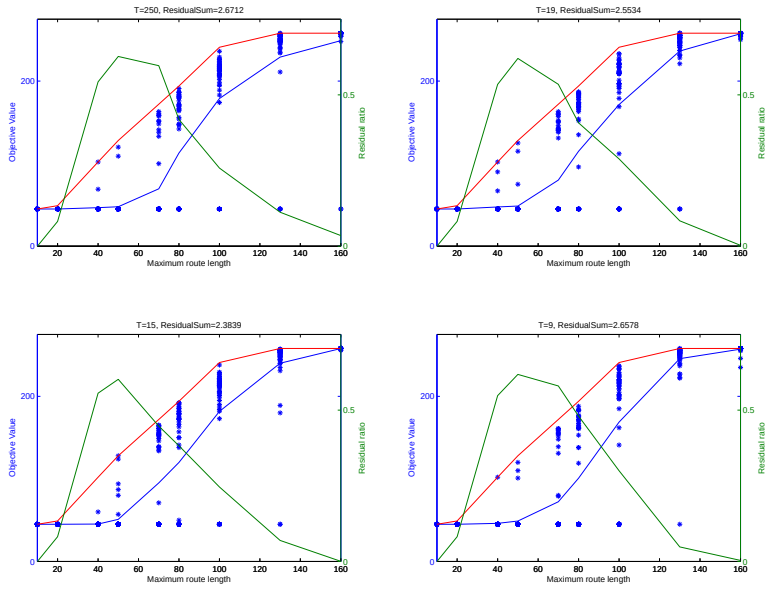



Figure 4.4: Shows results for four different temperatures. Blues line and dots is the average value and the calculated objectives, the red line is the best known objective and the green line is the residual ratio.

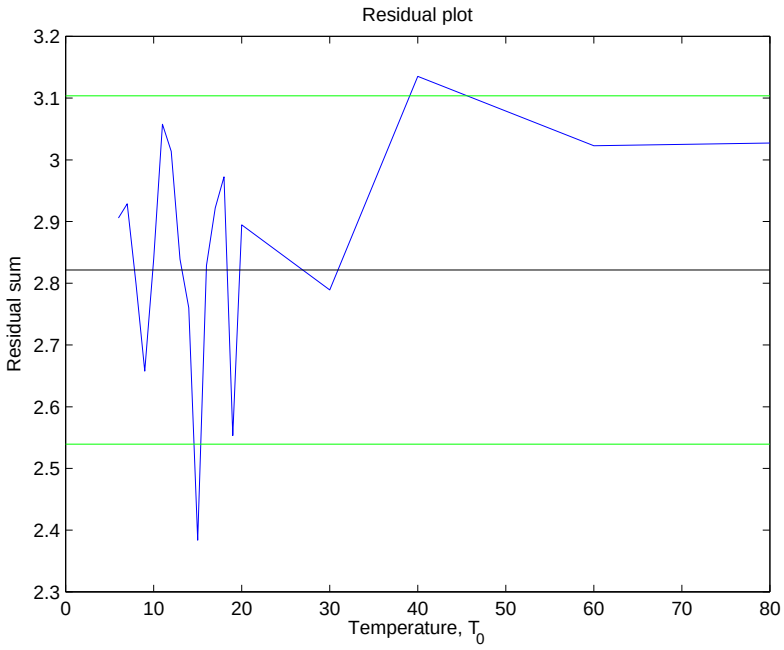



Figure 4.5: Shows the residual sum for some temperatures, blue line. The black line is the average residual sum for temperatures, $T_0 \in [6; 20]$, green dots are mean$\pm$10%.

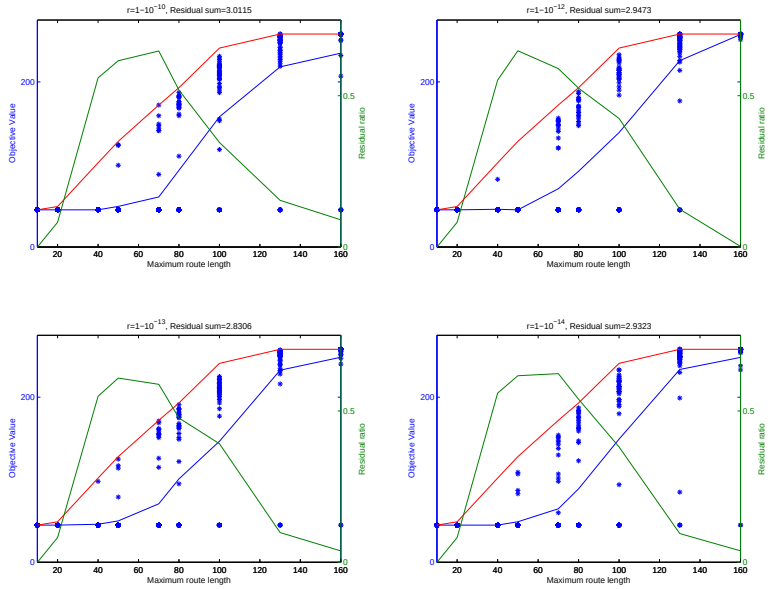



Figure 4.6: Shows results for four different temperatures. Blues line and dots is the average value and the calculated objective values, the red line is the best known objective and the green line is the residual ratio.

is $r = 0.5$. One can see, from Figure 4.7, that it gives worse results than other instances of reduction factor.
The reduction factor is now set to $r = 1 - 10^{-13}$ until a better value of $r$ is found.

### Results for Stopping Criteria, $F$

All results from the test can be seen in appendix **??**. The four best results can be seen in Figure 4.8 and the single best result found is the residual sum of 2.4320 when $F = 2$. Again though the random factors in the algorithm led to different results. If not then values calculated for $F = 1$ should have given the same results as the ones calculated previously, when ideal temperature and reduction factor were determined. This was not the case.

In Figure 4.9 a plot of the different residual sum compared with the different instances can be seen. The instances are: $F \in \{10, 8, 6, 5, 4, 3, 2, 1, \}$ and $F \in \{10^{-1}, 10^{-2}, 10^{-3}, ...10^{-9}, 10^{-10}\}$. In Figure 4.9 there is a drop in the values of residual sums in instances 5 and up. This means that $F \leq 4$ are better stopping criteria than $F > 4$. Therefore the average value for instances higher than 5 was inspected. As can be seen, from the plot, the values, of residual sum, are very similar for instances higher than 5, $F \in \{2, 1, 0^{-1}, 10^{-2}, 10^{-3}, ...10^{-9}, 10^{-10}\}$.

The stopping criteria is now set to $F = 2$ until otherwise specified. This means that when the tamperature reaches the value F a srict local search will begin, ending the possibility of accepting worse solutions. Now better values for all three parameters of the cooling schedule

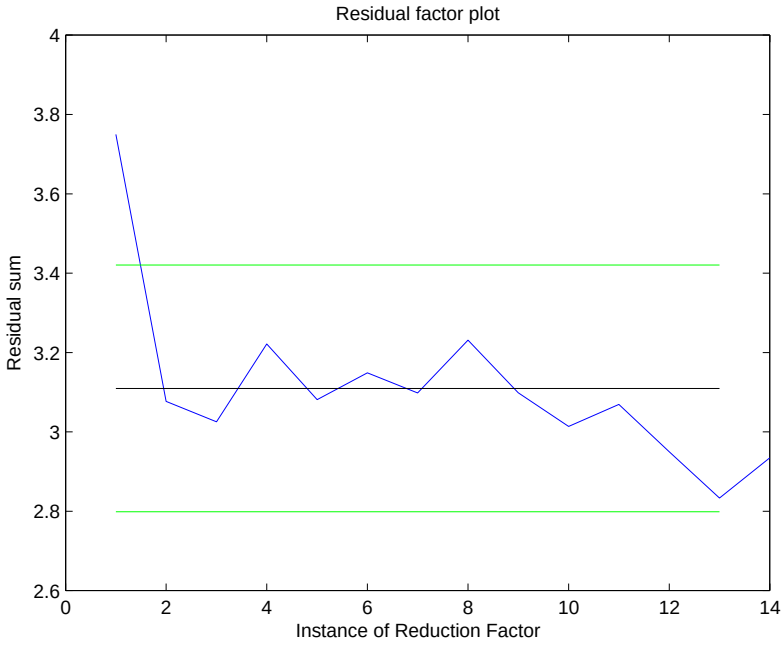



Figure 4.7: Shows the residual sum for all reduction factors, blue line. The black line is the average residual sum for all instances, green lines are mean±10%.

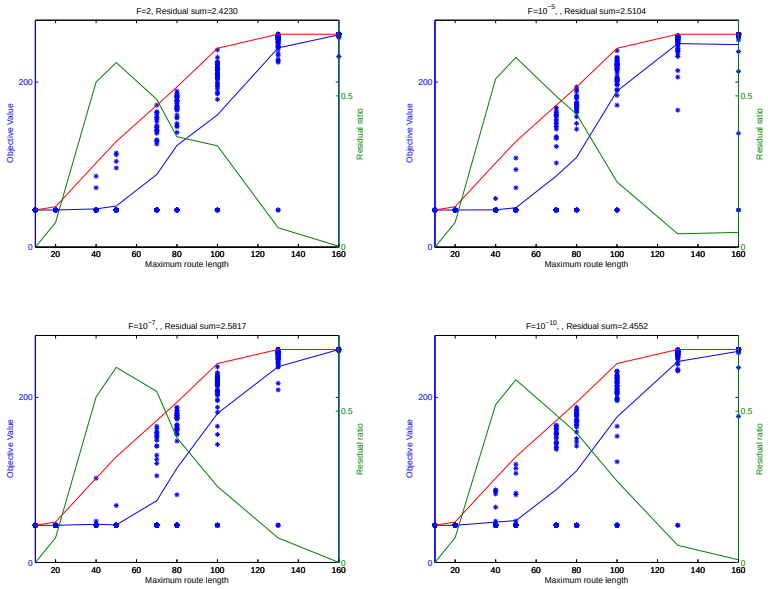



Figure 4.8: Shows results for four different temperatures. Blues line and dots is the average value and the calculated objectives, the red line is the best known objective values and the green line is the residual ratio.

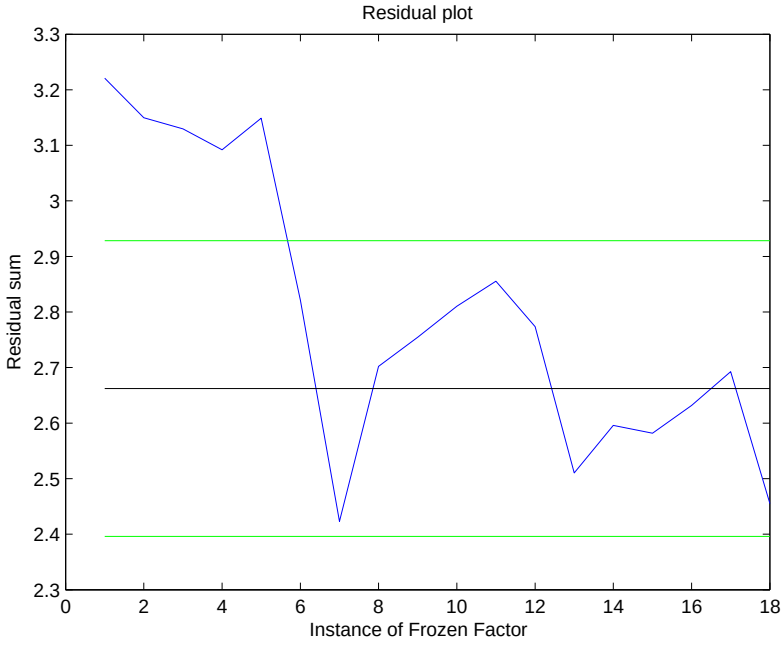



Figure 4.9: Shows the residual sum for all frozen factors, blue line. The black line is the average residual sum for instances 6 to 18, green lines are mean±10%.

have been identified. This schedule is $T_0 = 15$, $r = 1 - 10^{-13}$ and $F = 2$.

### 4.2.2 Second Trials for Calculating a Cooling Schedule using Data Set 50a

After a better cooling schedule had been determined a second run was constructed. This was done to retune the schedule with the new values. In these trials the same set for $M$ was used but a 100 runs were done for each value, of $M$. *The updated simulated annealing* and Decrease.java were both used.

As previously, a new, set of possible $T_0$, was constructed and tested on data set 50a. The reduction factor was set to $1 - 10^{-13}$ and the stopping criteria set to 2, both calulated values. The values of $T_0$ tested were 33 and ranged from 10000 to 6.

After the finding a good initial temperature a new reduction factor was calculated. The new $T_0 = T_0^*$ was used and the stopping criteria $F = 2$. There were 13 different values of $r$ tested raging from 0.99 to $1 - 10^{-14}$.

Next to find good stopping criteria a test was performed inspecting different values of the factor called frozen. In this inspection the better values of $T_0 = T_0^*$ and $r = r^*$ were used. There were 17 values inspected ranging from 4 to $10^{-10}$.

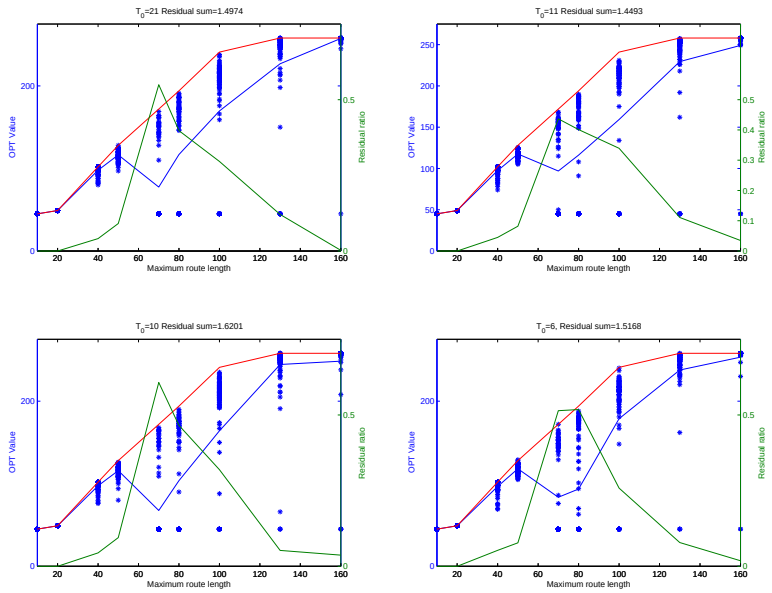



Figure 4.10: Shows results for four different temperatures. Blues line and dots are the average value and the calculated objective values, the red line are the best known objective values and the green line is the residual ratio.

### Results for Initial Temperature, $T_0$, using Data Set 50a

In the second run for inspecting possible values for $T_0$, each combination of $M \in \{10, 20, 40, 50, 70, 80, 100, 130,$ and $T_0 \in \{6, 7, 8, ..., 32, 33\}$ ran 100 times. This was done to limit the random factors in the algorithm. The four best results are shown in Figure 4.10. Note that individual results from this test can be seen in the appendix $D$.2.4.

The residual sum for all values of $T_0$ was then plotted in Figure 4.11. The best value according to this test was $T_0 = 6$, the old value $T_0 = 15$ is marked on Figure 4.11 with a red dot. In the first trials $T_0 = 6$ gave a rather high residual sum, see Figure 4.5, but increased number of test for each combination should limit the effects of random factors. In the previous test $T_0 = 9$ and $T_0 = 19$ gave good results, in this test both initial temperatures perform better than average. This can be seen by inspecting the black line displaying the average results, for all $T_0$, in Figure 4.11.
Individual result for each possible $T_0$ can be seen in the appendix.

The initial temperature is now set to $T_0 = 6$ until otherwise specified.

### Results for Rreduction Factor, $r$, using Data Set 50a

The second run for the reduction factor gave different result than the first one. The value of $r$ giving the lowest residual ratio sum was $r = 1 - 10^{-5}$. The four best results are shown in Figure 4.12. The four results are the four highest values, of reduction factor, tested.

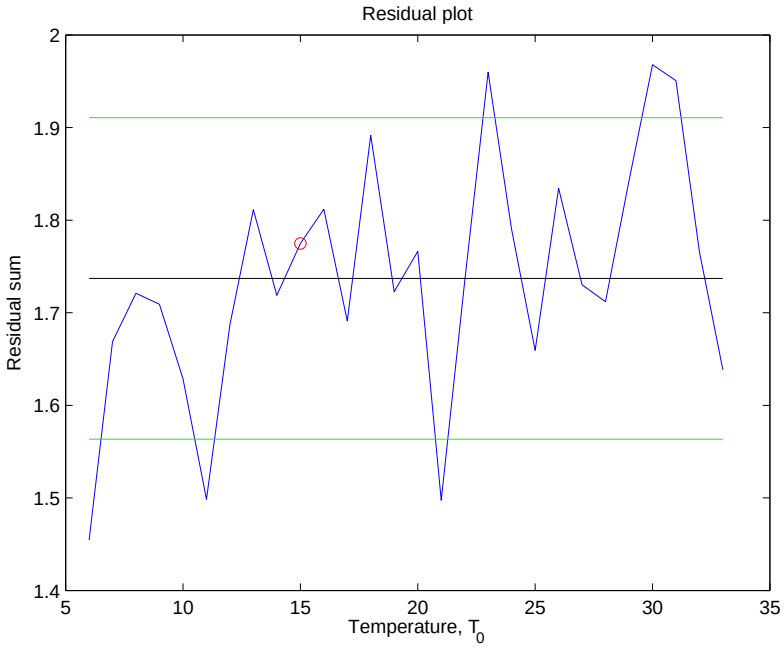



Figure 4.11: This figues shows the residual sum for some temperatures, blue line. The black line is the average residual sum for temperatures and green line is the mean±10%. The red dot is the best result from the previous test.

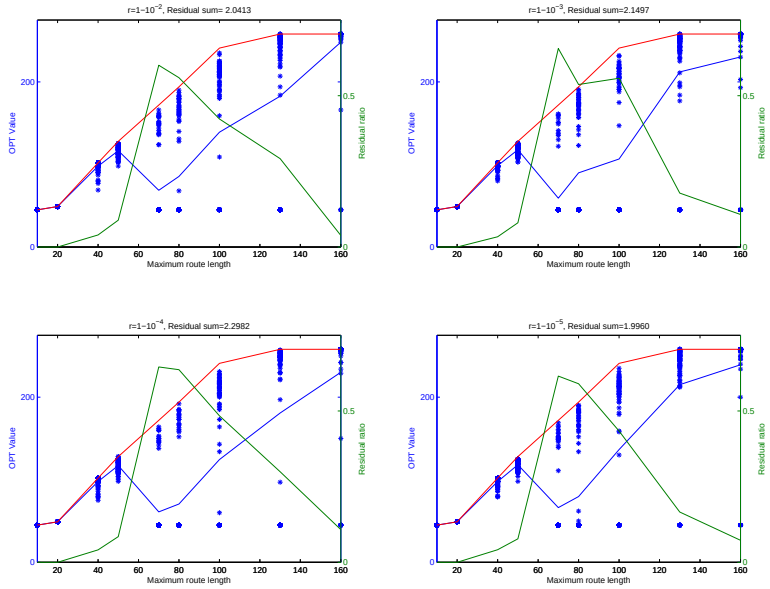



Figure 4.12: Shows results for four different temperatures. Blue line and dots are the average value and the calculated objective values, the red line are the best known objective values and the green line is the residual ratio.

Let us now inspect Figure 4.13. As previously the plot is of instance of reduction factors versus the residual sum. The only instance removed from the trial set was $r = 0.5$ as it gave the worst results. Therefore the instances are $\{1-10^{-2}, 1-10^{-3}, 1-10^{-4}, ...1-10^{-13}, 1-10^{-14}\}$. The plot 4.13 shows that the highest values tested resulted in the lowest sum of residual ratio. Compared to the previous test, of reduction factor, $r = 1-10^{-10}$ is the only of the previous top four results to preform better than average. Note that all other results from this test can be seen in the appendix $D$.2.5.

The reduction factor is now set to $r = 1-10^{-5}$ until otherwise specified.

### Results for Stopping Criteria, $F$, using Data Set 50a

The four best results, the ones with the lowest residual sum, can be seen in Figure 4.14 and the single best result found is the residual sum of 1.9757 when $F = 1-10^{-6}$, this is the result displayed in the bottom right courner of Figure 4.14. Other results from this test can be seen in the appendix $D$.2.5.

In Figure 4.15 a plot of the different residual sum compared with the different instances can be seen. The instances are: $F \in \{4, 3.5, 3, 2.5, 2, 1.5, 1, 0.58, \}$ and $F \in \{10^{-1}, 10^{-2}, 10^{-3}, ...10^{-9}, 10^{-10}\}$. In this plot, Figure 4.15, it is apperant that there are two local minimum values and of those one is the global minimum value. This global minimum value is found when $F = 10^{-6}$. In the

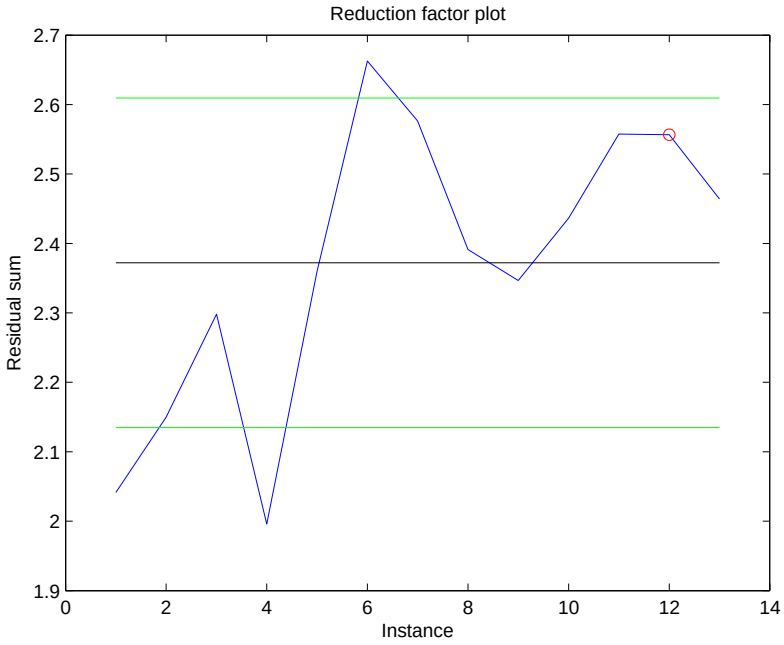



Figure 4.13: Shows the residual sum for all reduction factors, blue line. The black line is the average residual sum for all instances, green lines are mean±10%.

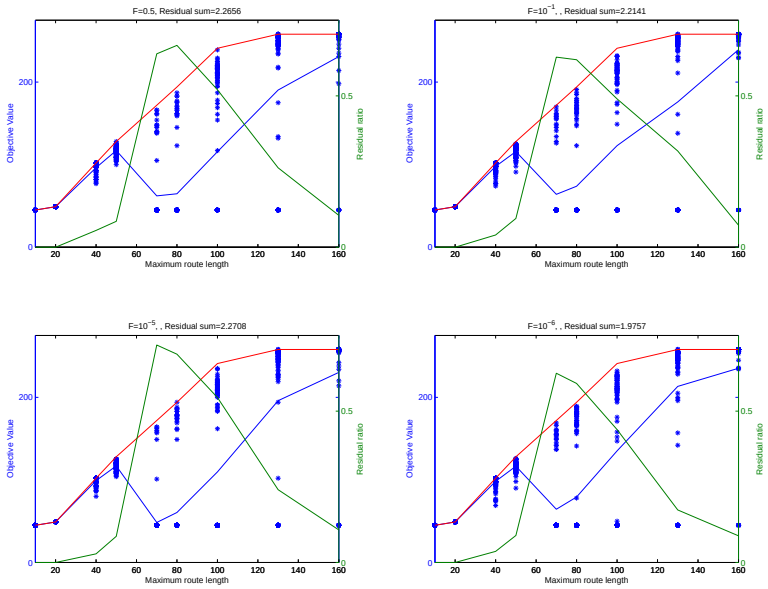



Figure 4.14: Shows results for four different temperatures. Blue line and dots are the average value and the calculated objective values, the red line are the best known objective values and the green line is the residual ratio.

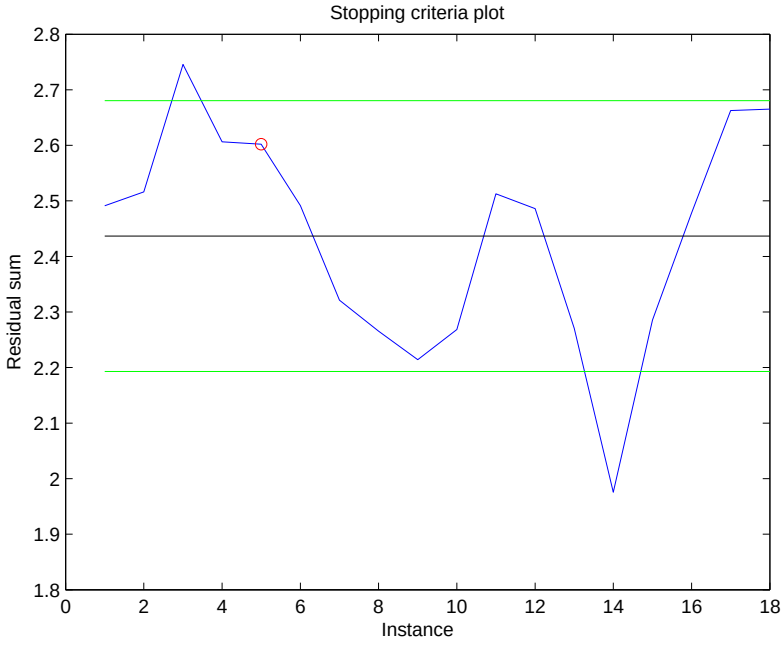



Figure 4.15: Shows the residual sum for all frozen factors, blue line. The black line is the average residual sum for all instances, green lines are mean±10%.

first trials $F = 10{-}6$ also gave good results.

The frozen factor is not set as $F = 10^{-6}$ until otherwise specified.

### 4.2.3 Cooling Schedual Trials for Data Set $3\_50\_a$

To ensure that the cooling schedule calculated is the best one available a new data set was used in cooling schedule experiments. This data set was $3\_50\_a$. After a new value $T_0$ or $r$ had been calculated that same value was used to determine the remaining values of $r$ or $F$. The set maximum route lengths was $M \in \{10, 20, 40, 50, 70, 80, 100, 130, 160\}$. Each combination of the tested factor ($T_0$, $r$ or $F$) and $M$ was run 50 times and each run had 50,000 iterations. The values in the objective function where $\alpha = 1$ and $\beta = 15$.

The results from tests with $3\_50\_a$ were only compared to the results from the second trials of test using data set 50a, as that cooling schedule was considered a better schedule because it had been retuned.

#### Results for Cooling Schedual Trials for Data Set $3\_50\_a$

Individual results for each trial, inspecting each factor ($T_0$, $r$ and $F$) can be seen in appendixes $D.2.6$, $D.2.7$ and $D.2.8$.

The cooling schedule should return good results for any number of data sets. Therefore the

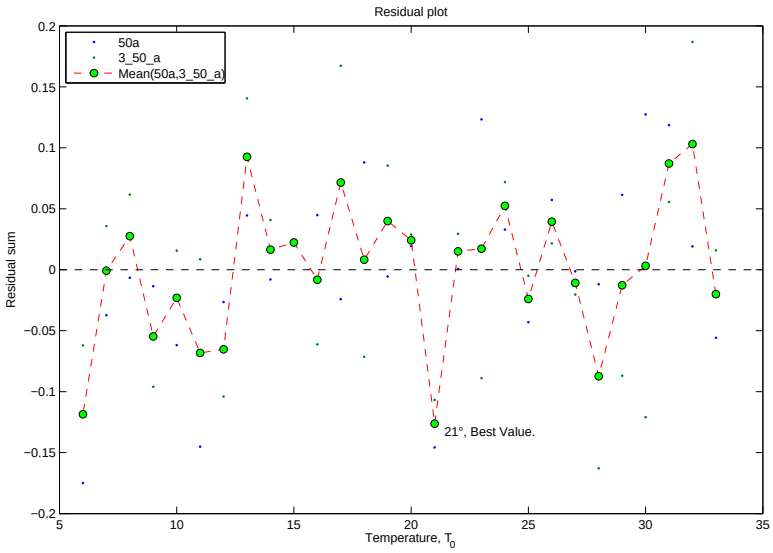



Figure 4.16: Shows the scaled residuals ratios for different temperatures for data sets 50a and 3_50_$a$.

results from the second trials using data set 50a and the trials using data set 3_50_$a$ were compared. It is known that data set 3_50_$a$ is simpler, or its routes are easier to construct, than data set 50a. This leads to lower residuals sums for data set 3_50_$a$. To compare the results the two the residuals needed to be scaled. This is done in a few steps. First let us define the set of residuals as A (for 50a) and B (for 3_50_$a$). To even things out a little bit a logaritmic function calculated for each set, $ln(A)$ and $ln(B)$. This gives equal importance to the high values residuals calulated with data set 50a and the high value residals calculated with 3_50_$a$. After this is done the average value of each logarithmic set, $m_{ln(A)}$ and $m_{ln(B)}$, were withdrawn from the set. This ensures that both sets are distributed around zero. The final values of residual sums expolered were $ln(A) - m_{ln(A)}$ and $ln(B) - m_{ln(B)}$.
The results from all this can be seen in Figure 4.16. This figure shows that the best starting temperature, $T_0$, for both of these data sets is $T_0 = 21$. This temperature was then used to determinie the best $r$ and $F$, the two remaining tests in this section.

The same method was then applied to the sets of results from different reduction factors. The scaled residual plot from that can be viewed in Figure 4.17. From that plot it is seen that instance number 4 gives the best reduction factor for these two data sets. The instances are the same as in the second trials with data set 50a, meaning that the best reduction factor is $r = 1 - 10^{-5}$. This is the same result as the second trials with data set 50a determined. The reduction factor used to determine $F$ is r=$1 - 10^{-5}$.

The results from tests, used to determine the frozen factor, were scaled as previously explained. Theses scaled results can be seen in Figure 4.18. The figure shows the best rusults found at instance 14 were $F = 10^{-6}$, which is the best stopping criteria for data sets 50a and 3_50_$a$.

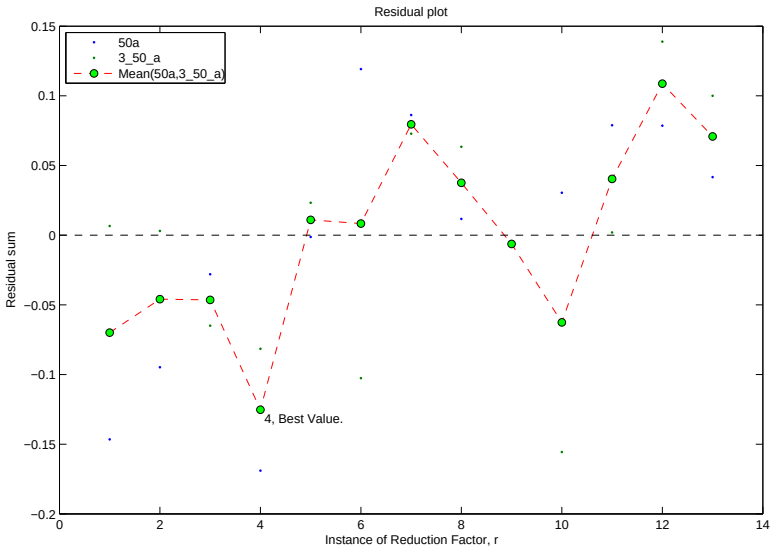



Figure 4.17: Shows the scaled residuals ratios for different reduction factors for data sets 50a and 3_50_$a$.

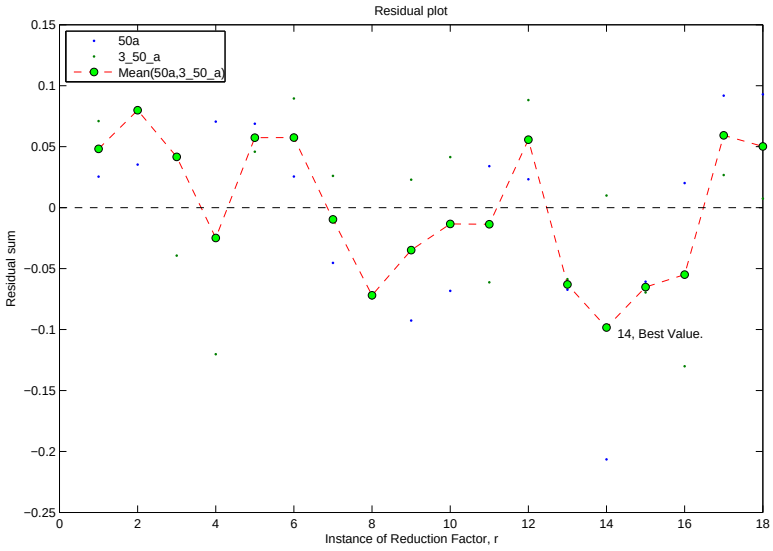



Figure 4.18: Shows the scaled residuals ratios for different frozen factors for data sets 50a and 3_50_$a$.

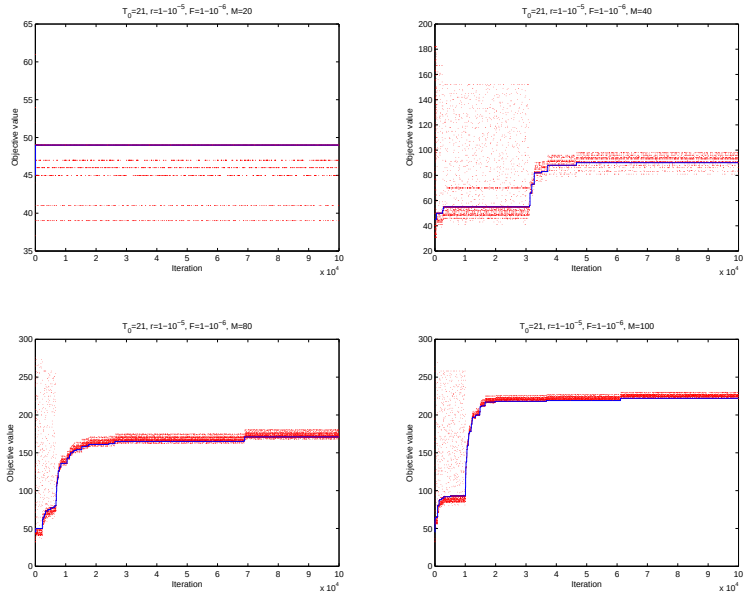



Figure 4.19: Shows the change in objective function in each iteration step, blue line. Also the proposed solution, red dots, is also shown.

In Figure 4.19 the objective value and suggested solutions can be observed. The cooliing schedule used in Figure 4.19 is $T_0 = 21$, $r = 1 - 10^{-5}$ and $F = 10^{-6}$ and the data set is 50a, the number of iterations was 100,000.

The best cooling schedual found for data sets 50a and 3_50_$a$ is $T_0 = 21$, $r = 1 - 10^{-5}$ and $F = 10^{-6}$.

## 4.3 Determining the Probability Matrix

In the class moves.java it is determined what action will be taken in the current iteration. The actions are defined as moves[4]. Now in each iterations there are different scenarios. In one scenario there might be no unused routes and therefore it would be impossible to call BusMove.java, in another there might be no unused points left then one cannot call an insert move or a bus move. This means that in some scenraios one is only possible to use certain moves, while in other scenarios one might be able too call all moves. If a move that is infeasible to use in a certain scenario is called, in that paticular scenario, the result will be a calculation error in the algorithm. Overall there are seven scenarios and in each case, except one[5], there are different odds for different moves. To represents these odds a matrix, $P$, was constructed. Where:

[4] InsertMove11.java, InsertMove13.java, SwapMove11.java, SwapMove21.java, SwapMove31.java and BusMove11.java

[5] In scenario 5 one can only call SwapMove11.java

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} \\ p_{41} & p_{42} & p_{43} & p_{44} & p_{45} & p_{46} \\ p_{51} & p_{52} & p_{53} & p_{54} & p_{55} & p_{56} \\ p_{61} & p_{62} & p_{63} & p_{64} & p_{65} & p_{66} \\ p_{71} & p_{72} & p_{73} & p_{74} & p_{75} & p_{76} \end{pmatrix}$$

The scenarios are:

| **Scenario** | **Description** |
|---|---|
| 1 | More the one route in use, but not all, and no available points. |
| 2 | All buses in use and available points. |
| 3 | All buses in use and no available points. |
| 4 | More the one route in use, but not all, and available points. |
| 5 | One route and no available points. |
| 6 | One route and available points. |
| 7 | No routes and available points. |

The numbers of the scenarios reflect the way they were programmed and is not important in any other way. It should be noted that after Decrease.java was constructed a new scenario was discovered. This would be referred to as scenario 8, in that case no routes are active and no unused points are available. In this scenario the only possible solution is the empty solution giving the objective value as $\beta|K|$.

The moves are:

| **Number** | **Type of move** |
|---|---|
| 1 | SwapMove21.java |
| 2 | SwapMove11.java |
| 3 | InsertMove11.java |
| 4 | BusMove.java |
| 5 | SwapMove31.java |
| 6 | InsertMove13.java |

So in some scenarios it is impossible to call certain actions, moves. Therefore to update $P$ these indesable moves are replaced with zero, in the matrix.

$$P = \begin{pmatrix} p_{11} & p_{12} & 0 & 0 & p_{15} & 0 \\ p_{21} & p_{22} & p_{23} & 0 & p_{25} & p_{26} \\ p_{31} & p_{32} & 0 & 0 & p_{35} & 0 \\ p_{41} & p_{42} & p_{43} & p_{44} & p_{45} & p_{46} \\ 0 & p_{52} & 0 & 0 & 0 & 0 \\ 0 & p_{62} & p_{63} & p_{64} & 0 & p_{66} \\ 0 & 0 & p_{73} & p_{74} & 0 & 0 \end{pmatrix}$$

When test on the matrix, $P$, are conducted there is no need to look at line number 5 as:

$$\sum_{j=1}^{6} p_{ij} = 100 \quad \forall i \in \{1, 2, 3, 4, 5, 6, 7\}$$

Therefore $p_{52} = 100$ in all test.

$$P = \begin{pmatrix} p_{11} & p_{12} & 0 & 0 & p_{15} & 0 \\ p_{21} & p_{22} & p_{23} & 0 & p_{25} & p_{26} \\ p_{31} & p_{32} & 0 & 0 & p_{35} & 0 \\ p_{41} & p_{42} & p_{43} & p_{44} & p_{45} & p_{46} \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & p_{62} & p_{63} & p_{64} & 0 & p_{66} \\ 0 & 0 & p_{73} & p_{74} & 0 & 0 \end{pmatrix}$$

Before the test are started an initial guess for $p$ must be constructed:

$$P_1 = \begin{pmatrix} 40 & 40 & 0 & 0 & 20 & 0 \\ 30 & 30 & 20 & 0 & 10 & 10 \\ 40 & 40 & 0 & 0 & 20 & 0 \\ 30 & 20 & 25 & 10 & 10 & 5 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 30 & 50 & 10 & 0 & 10 \\ 0 & 0 & 90 & 10 & 0 & 0 \end{pmatrix}$$

The matrix, $P_1$, was constructed entirely with logical guess as no other methods were available. This was done by guessing what moves would be important in each scenario. For example if there are no unused points left it is a given that *swap moves* are important. Of these three *swap moves* 11 and 21 work faster than *swap move 13* this is because the former two move two nodes while the latter only moves one node. Therefore the odds in scenario one feature an equal possibilty of choosing 11 or 21 but a slightly smaller possibility of choosing 31. This kind of logic was implemented for all scenarios. The scenarios are independent of each other in any single iteration. They may though be connected when many iterations are put together.

In order to find the best odds for each action, move, each scenario will be looked at separately[6]. So in order to find the best odds in line 3, in $P$, the rest of the matrix will be locked in the initial guess while all possibilities for line 3 are inspected. The same was then done for all other lines of the matrix, $P$.

There were two tests constructed to determine the different values of $P$ for different data sets. The data set used for first experiment was 3_50_$a$ with $M = 20$, $T_0 = 3000$, $r = 0.5$, $F = 0$ and there were 10 trials for each possibility. The maximum number of routes was 3.

The second data set used was 50a with $M = 100$, $T_0 = 3000$, $r = 0.5$, $F = 0$, and there were 10 trials for each possibility. The maximum number of routes was 3. Note the in this test, using this data set, a second test was conducted for scenario one. $M = 160$ and the maximum number of buses was 5 for the second test of scenario one. The reason for this is that the scenario cannot occur unless all the points are visited without using all the routes.
Note that the first test, involving scenario one and data set 50a, used $P_1$ to try to determine the best value of $P$, for this scenario, but the results were not usable. Therefore a second test using $P_2$, as a reference while inspecting possible values, was conducted and gave usable results. The value of $P_2$ is defined in the results section.

[6]Looking at all the scenarios together may is to complicated as there are to many possibilities.

### 4.3.1 Results for Data Set 3_50_$a$

To limit the running time, of the tests, the probabilities did not run on a single percent, $\{1\%, 2\%, 3\%\ldots.\}$, in line 3 alone there would be over 125,000 possibilities[7]. Rather on ten percent,$\{0\%, 10\%, 20\%, 30\%\ldots.\}$. This decreased calculation time enough to run the tests.

| Scenario | Objective Value | Iteration | Travel Time |
|---|---|---|---|
| 1 | 231.3 | 26,391 | 57.6083 |
| 2 | 231.6 | 22,061 | 57.5682 |
| 3 | 229.2 | 28,904 | 58.1504 |
| 4 | 231.4 | 31,025 | 57.7911 |
| 6 | 231.4 | 29,622 | 58.2504 |
| 7 | 229.1 | 28,963 | 57.4555 |
| | **232** | | |

The table above shows the average objective value, the average iteration where the objective value was found and the total travel time for all the routes. In the last line of the table one can see the best known objective value. This test then gave a new probability matrix, $P_2$, viewable below:

$$P_2 = \begin{pmatrix} 50 & 30 & 0 & 0 & 20 & 0 \\ 0 & 10 & 10 & 0 & 30 & 50 \\ 60 & 30 & 0 & 0 & 10 & 0 \\ 0 & 10 & 10 & 60 & 20 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 & 30 \\ 0 & 0 & 70 & 30 & 0 & 0 \end{pmatrix}$$

The probability matrix[8] $P_2$ was constructed by inspecting not nearly all possible solutions, as that would have taken to long[9]. Therefore to inspect if single percentage values are important $P_2^*$ was constructed. The test was very similar to the one used to construct $P_2$ but implemented an updated simulated annealing. The solutions close to the one calculated in $P_2$ were the only ones inspected.

| Scenario | Objective Value | Iteration | Travel Time |
|---|---|---|---|
| 1 | 231.9 | 19,956 | 56.9000 |
| 2 | 231.8 | 25,712 | 57.0000 |
| 3 | 231.9 | 22,506 | 56.9000 |
| 4 | 232 | 18571 | 57.0000 |
| 6 | 231.8 | 21,686 | 56.8000 |
| 7 | 231.8 | 23,397 | 56.5000 |
| | **232** | | |

The changes in average objective a relatively small, can be contributed to the random factor in the algorithm[10], for most scenarios. Although the greatest changes seen in average objective values are found in scenarios 3 and 7.

[7] This would then have a runtime of more than a 1000 hours, given an average runtime of about 30 seconds
[8] The probability matrix used in tests for the cooling schedule was $P_2$.
[9] Approximately 120 days, at the time.
[10] All nodes are inserted randomly into the soltuion.

$$P_2^* = \begin{pmatrix} 53 & 34 & 0 & 0 & 13 & 0 \\ 0 & 7 & 10 & 0 & 25 & 58 \\ 59 & 34 & 0 & 0 & 7 & 0 \\ 2 & 7 & 6 & 55 & 16 & 14 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 72 & 0 & 19 \\ 0 & 0 & 74 & 26 & 0 & 0 \end{pmatrix}$$

It is necessary for each line of the probability matrix to have a total sum of 100%. Therefore when the second trials, determining $P_2^*$, were consructed a fail safe was put into the test algorithm to ensure constraint was met. Therefore some values in $P_2^*$, and later in $P_3^*$, differ more than 5% from there conterparts in $P_2$ and $P_3$.

### 4.3.2 Results for Data Set $50a$

The result in the table below show the objective value calculated, the iteration the value was found and the combined travel time for all the routes.

| Scenario | Objective Value | Iteration | Travel Time |
|---|---|---|---|
| 1 | 105.6 | 21294.1 | 107.73 |
| 2 | 105.8 | 18187 | 118.00 |
| 3 | 121.2 | 23580.9 | 145.23 |
| 4 | 125.7 | 22051.9 | 146.05 |
| 6 | 96.6 | 19821.3 | 106.36 |
| 7 | 213.8 | 28945.6 | 294.83 |
| | **241** | | **293.24** |

When the table above is inspected it is obvious that the probability values for scenario 7 are the most important. Big changes in those values result in a much higher average calculated objective value. The results in other scenarios are much lower the best known objective, shown in the bottom line. The value of $P = P_3$ is shown below.

$$P_3 = \begin{pmatrix} 30 & 70 & 0 & 0 & 0 & 0 \\ 10 & 50 & 20 & 0 & 10 & 10 \\ 0 & 10 & 0 & 0 & 90 & 0 \\ 0 & 50 & 0 & 20 & 10 & 20 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 60 & 0 & 20 & 0 & 20 \\ 0 & 0 & 20 & 80 & 0 & 0 \end{pmatrix}$$

The results from the test used to re-evaluate scenario one are seen in the table below. Remember that this test used $P_2$ as an inital guess not $P_1$.

| Scenario | Objective Value | Iteration | Travel Time |
|---|---|---|---|
| 1 | 262.5 | 21384 | 263.4 |
| | **273** | | |

$$p_{1,i} = \begin{pmatrix} 70 & 10 & 0 & 0 & 20 & 0 \end{pmatrix}$$

So according to this the value of $P$ best suited for solving problem defined with data set 50a is:

$$P_3 = \begin{pmatrix} 70 & 10 & 0 & 0 & 20 & 0 \\ 10 & 50 & 20 & 0 & 10 & 10 \\ 0 & 10 & 0 & 0 & 90 & 0 \\ 0 & 50 & 0 & 20 & 10 & 20 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 60 & 0 & 20 & 0 & 20 \\ 0 & 0 & 20 & 80 & 0 & 0 \end{pmatrix}$$

A second trial was done to determine the single percent values of $P_3$. The new matrix constructed was called $P_3^*$. These new trials gave the following results. Note though that an updated version of the simulated annealing algorithm was used along with Decrease.java and $P_3$ was used as the old matrix when evaluating $P_3^*$. All other factors are the same as when $P_3$ was calculated.

| **Scenario** | **Objective Value** | **Iteration** | **Travel Time** |
|---|---|---|---|
| 1 | 258 | 48008 | 417.9 |
| | **273** | | |
| 2 | 220.8 | 35292 | 293.7 |
| 3 | 216.1 | 30787 | 294 |
| 4 | 222.6 | 32150 | 296.2 |
| 6 | 217.9 | 37688 | 291.9 |
| 7 | 216.6 | 36484 | 294.1 |
| | **241** | | **293.24** |

$$P_3^* = \begin{pmatrix} 72 & 5 & 0 & 0 & 23 & 0 \\ 7 & 46 & 18 & 0 & 7 & 22 \\ 0 & 24 & 0 & 0 & 76 & 0 \\ 3 & 452 & & 17 & 7 & 26 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 55 & 0 & 22 & 0 & 23 \\ 0 & 0 & 18 & 82 & 0 & 0 \end{pmatrix}$$

The two matrixies $P_2$ and $P_3$ were compared. This is shown in later sections. Of the two $P_2$ provided better results and was then compared to $P_2^*$. Because $P_3$ provide worse results $P_3^*$ was not inspected further.

## 4.4 Exploring Different Insert Moves

In this project there were four *insert moves constructed* 11, 12, 14 and 15. Another *insert move*, called 13, is also used but is differant from the other four as it removes a node before inserting a new one. Therefore 13 will not be put in the same catagory as 11, 12, 14 and 15. Tests using data set 3_50_$a$ were preformed 10 times for each combination of insert move and

$M \in \{5, 10, 15, 20, 25, 30, 35, 40, 50\}$. For each test several values were inspected. First the objective value was inspected, it was the most important value in the test. If the best known objective value, the objective calculated by hand, was reached then for $M \in \{5, 10, 15, 20, 25, 30, 35\}$ the route was also inspected, for some $M$ to find maximum profit the algorithm had to find the best route, this can also be calculated due to the structure of the dat set. For $M \in \{40, 50\}$ the maximum profit could be achieved without using the best route. The iteration value when the objective value was reached was inspected. All tests preformed 50,000 iterations, the initial temperature was set to 11, $T_0 = 11$, the reduction factor was set at $1-10^{-13}$, $r = 1-10^{-13}$ and frozen factor at 2, $F = 2$. Also an updated version of simulated annealing and Decrease.java were implemented in these tests. In the objective function $\alpha = 1$ and $\beta = 15$, the beta value was chosen such that more than one node had to be added to a new route. The probability matrix $P$ was set to $P_2$.

Note that all factors in the program remained unchanged except *insert moves* 11, 12, 14 and 15; when these experiments were carried out. One could still choose a *swap move* if the algorithm and the scenario at hand demanded it.

### 4.4.1 Results

Let $OPT$ be the returned objective given by the program. Let $z$ be the best known objective value, calculated by hand, for the given problem. Then define:

$$Z = \sum_{i=1}^{N} \frac{OPT_i}{N} \quad N \text{ are the number of runs for a given M.} \tag{4.4.1}$$

So the Table 4.3 shows $z/Z$ for all M and the four *insert moves*.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| InserMove11 | 1 | 0.99009 | 0.94678 | 0.99914 | 0.99302 | 0.95651 | 0.96983 | 0.95579 | 0.98611 | 0.97747 |
| InserMove12 | 1 | 0.77477 | 0.74854 | 0.72414 | 0.71429 | 0.78947 | 0.78589 | 0.91111 | 0.93796 | 0.82069 |
| InserMove14 | 1 | 0.67568 | 0.88246 | 0.92672 | 0.67409 | 0.70305 | 0.73942 | 0.90602 | 1 | 0.83416 |
| InserMove15 | 1 | 0.94595 | 0.99415 | 0.94828 | 0.91694 | 0.99723 | 0.85742 | 0.9375 | 0.97824 | 0.95286 |

Table 4.3: Shows the ratio between the returned objective value and the best known objective value calculated by hand for each $M$.

If Table 4.3 is inspected one can see that *insert move* 11 returns on average the best objective values. Also in most cases it has the highest ratio for any given $M$. although in two cases *insert move* 15 performed better. Of the four *insert move* 11 is the only one that performed well for all $M$, *insert move* 12 tends too give the worst performance. *Insert move* 12, shows poor results for most M, except when $M \in \{40, 50\}$. In those cases $M$ is greater than the maximum route length and *insert move* 12 returns acceptable results. The reason for these results may be that *insert move* 12 always chooses the highest profit node with the lowest node number. Therefore if two nodes had equal importance, say $node_{12}$ and $node_{41}$, then InsertMove12.java would always choose $node_{12}$.

Another *insert move* showed poor results for all M except those values that are longer than the longest possible route and this was *insert move* 14. In cases where $M = 50$ *insert move* 14 returns excellent results. This is because the method always chooses the node with the highest profit farthest from the depot. Therefore when this node is outside the maximum route length, *insert move* 14 cannot find any node to insert into the routes.

The only other *insert move* to return good results is *insert move* 15. It always chooses the highest profit node closest to the depot. Therefore InsertMove15.java always finds nodes to insert into its routes as long as there are high profit nodes inside the range of the maximum route. The main problem with InsertMove15.java is if there is a low profit node that one can fit into the route, then *insert move* 15 will not add that node to the route. Still the method did often produce acceptable results.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| InserMove11 | *21, 456* | 17,120 | 14,640 | 20,142 | 15,669 | 24,152 | 29,122 | 44,513 | 10,752 | **21,951** |
| InserMove12 | 24,513 | 13,202 | *3, 419* | *3, 297* | *5, 176* | *16, 720* | *22, 707* | 30,749 | 35,755 | **17,282** |
| InserMove14 | 49,038 | 29,027 | 17,481 | 18,776 | 24,930 | 37,997 | 37,664 | *22, 143* | *8, 654* | **27,301** |
| InserMove15 | 24,358 | *7, 716* | 5,327 | 6,578 | 11,703 | 34,121 | 34,174 | 55,155 | 24,703 | **22,648** |

Table 4.4: Shows the number of iterations used to find the returned objective value.

Table 4.4 shows the average number of iterations it took for the program to find the objective values returned for each value of $M$. The values are not that different but *insert move* 12 was fastest and *insert move* 14 used the most iterations to find an optimum value. The results show that *insert move* 11, the most effective of the four, does not require an extraordinary amount of iterations to return an objective value.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| InserMove11 | *21, 456* | 17,129 | 15,699 | 21,375 | *16, 075* | *24, 342* | *30, 203* | 62,276 | 47,628 | **28,464** |
| InserMove12 | 24,658 | 13,631 | *5, 261* | 12,496 | 17,374 | 27,190 | 44,066 | 48,681 | 89,509 | **31,429** |
| InserMove14 | 49,227 | 31,193 | 18,560 | 20,985 | 25,934 | 43,857 | 42,236 | *40, 727* | *40, 715* | **34,825** |
| InserMove15 | 24,358 | *7, 716* | 5,465 | *7, 033* | 12,833 | 34,884 | 40,145 | 63,851 | 62,832 | **28,790** |

Table 4.5: Shows the number of iterations used to find the returned path.

Table 4.5 above shows the average number of iterations it took for the program to find the returned route for each value of $M$. Here *insert move* 11 and *insert move* 15 use the fewest iterations. This still shows that *insert move* 11 is an acceptable method when considering the amount of iterations needed to find a good route. Note that in most instances, especially when $M \in \{5, 10, 15, 20, 25, 30, 35\}$, the number of iterations needed to find an objective value and the number needed to find an returned route are often similar.

In Table 4.6 the number of cases, for each $M$, found were the best known objective was reached using the best known path are listed. If a certain run returned the best known objective value, calculated by hand, taking and the best known route the test was marked with a Boolean *true* if this was not the case then the run returned a Boolean $false$. Of the four *insert moves*

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| InserMove11 | 10 | 9 | 5 | 8 | 7 | 6 | 1 | 3 | 1 | 5.0 |
| InserMove12 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 1.3 |
| InserMove14 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1.2 |
| InserMove15 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1.2 |

Table 4.6: Shows the numer of times best know objective value is reached using best known path.

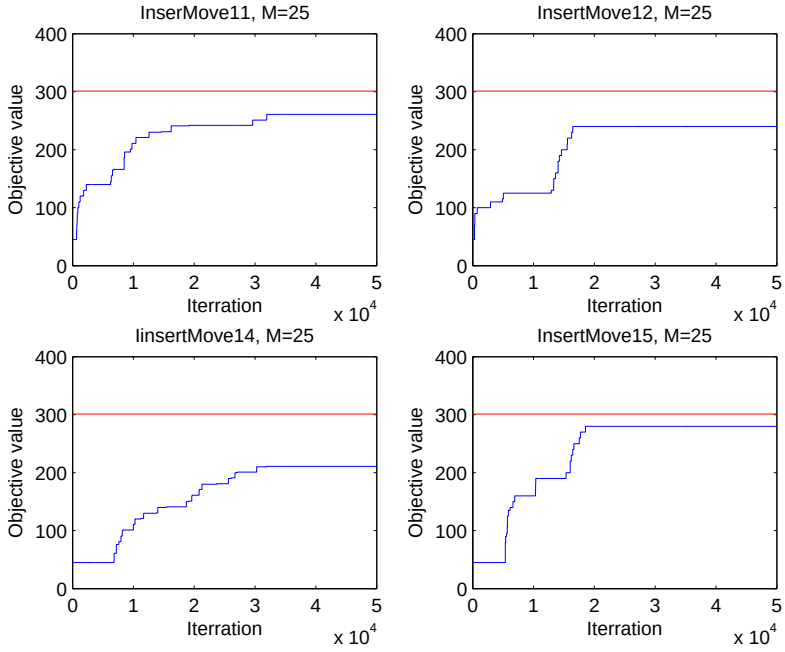



Figure 4.20: Shows a example of the objective function, for all insert moves, in each iteration, blue line. The red line represents the best known objective value.

the first, *insert move* 11, gives the highest values of *true*. The reason *insert move* 15 does not return good results, as it did with the objective value, because of the method it uses.

In Figure 4.20 one can see the objective value changing with regards to the iterations. In the figure $M$ is set to 25, note that none of the methods reach the best known objective value, the red line. Of the four methods *insert move* 15 is closest to the best known objective value and *insert move* 14 is furthest from the best known objective value.

In Figure 4.21 one can clearly see the difference between methods. The residual ratio is consistently low for *insert move* 11. Both *insert move* 12 and *insert move* 14 give bad results and *insert move* 15 acceptable results but not as good as *insert move* 11.

In conclusion one can see, from tables 4.3, 4.4, 4.5 and 4.6; and Figure 4.21, that of the four

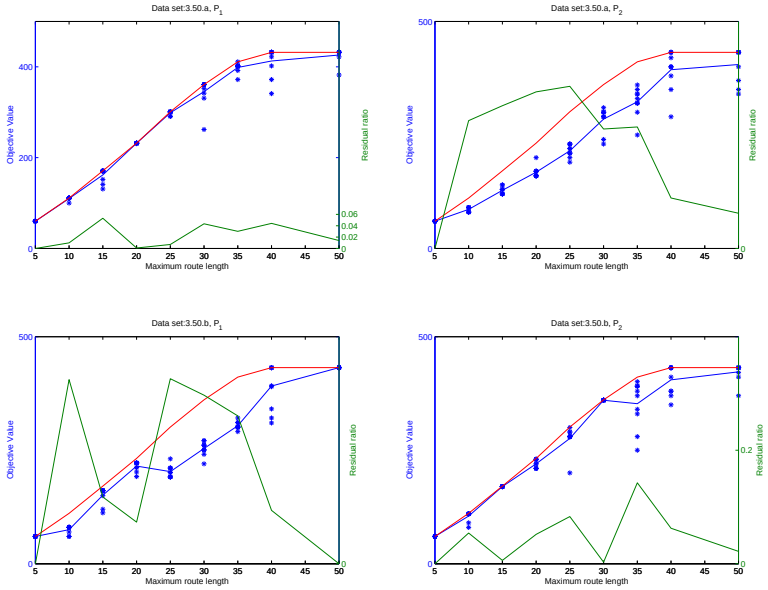



Figure 4.21: Shows calculated results in comparison with the best known objective value. The blue line is the average results, the blue dots are each calculated result and the red line is the best known objective value. The green line is the residual ratio and is represented on the right y-axis.

*insert move 11* gives the best results and does so in an acceptable amount of iterations.

From this one can see that the *insert move* 11, a random insertion into routes, returns the best results. Therefore it will be the only *insert move* used here after, unless otherwise specified.

## 4.5 Non-Randomly Generated Data Sets

The twelve non-randomly generated data sets were tested on the algorithm in this section. The cooling schedule used was $T_0 = 15$, $r = 1 - 10^{-13}$ an $F = 2$. Decrease.java was used along with the updated version of simulated annealing. The number of iterations for each test was 50,000 and the runs for each possible compnation of $M$ and $|K|$ was ten. In all tests using non-randomly generated data sets $\alpha = 1$ and $\beta = 15$ in the objective function. In the these data sets one could calculate the best known objective value by hand. This was possible because of the structure of the data sets.

### 4.5.1 Results Data Set 3_50_$a$

The test performed on data set 3_50_$a$ used $|K| = 3$ and $M \in \{5, 10, 15, 20, 25, 30, 35, 40, 50\}$. As the use of Decrease.java will remove points from the data set the number of points used for each $M$ is shown in Table 4.7, these points are called feasible. Note that the maximum number of points is 45, not 50, because 10% of points have been removed, to include an equal amount of nodes in each route the number of points could not be 47, $47/3 = 15.667$ and the depot then counts as an additional two points. Also in Table 4.7 are displayed the calculated best known objective values. These are known best objective values as the unique structure of the data sets allows one to calculated, by hand, the best objective values and paths.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|
| 3_50_$a$ | 10 | 22 | 33 | 41 | 45 | 45 | 45 | 45 | 45 |
| **Best** | 60 | 111 | 171 | 232 | 301 | 361 | 411 | 432 | 432 |

Table 4.7: Shows the number of points feasible for each $M$ and the best known objective value, calculated by hand.

In Figure 4.22 the structure of the data set is displayed.

To determine which probability mattices are the best it was necessary to compare them.

Table 4.8 shows the results from calculations using $P_1$, Table 4.9 shows results from calculations using $P_2$ and Table 4.10 shows results from calcualtions using $P_3$. If the best calculated values in the three tables (4.8, 4.9 and 4.10) are compared to the best known objective values, calculated by hand, in Table 4.7 one can see that the best known objective value is found by the algorithm in most cases. Of the three possabilities matrices $P_2$ returns the best solutions. Notice that neither $P_1$ or $P_3$ found the best known objective value when $M = 35$. This is because of the uniqueness of the solution, displayed in Figure 4.23. The solution for $M = 35$

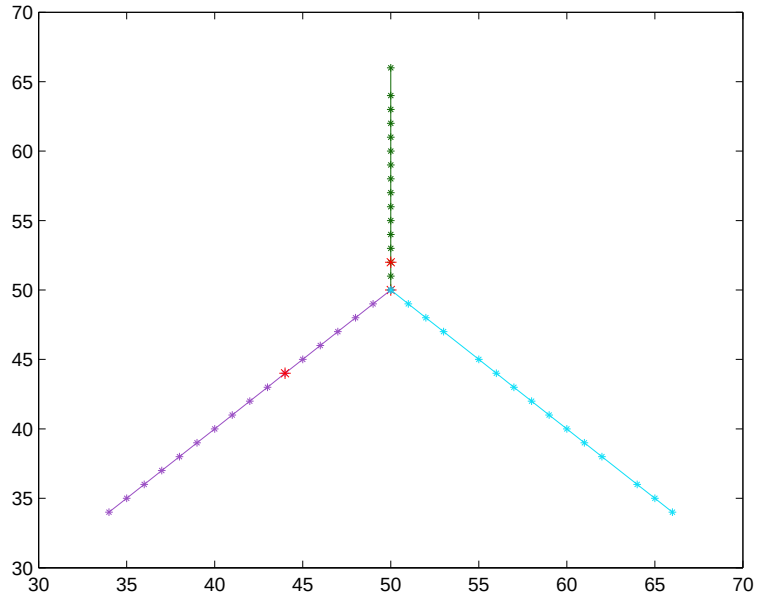



Figure 4.22: Shows the data set 3_50_$a$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|
| **Avrage** | 51.5 | 101.3 | 150.6 | 228.1 | 282.8 | 341.1 | 360.6 | 380.8 | 429 |
| **Best** | 60 | 111 | 171 | 232 | 301 | 361 | 401 | 432 | 432 |
| **Ratio** | 0.85833 | 0.91261 | 0.8807 | 0.98319 | 0.93953 | 0.94488 | 0.89925 | 0.88148 | 0.99306 |
| **CPU** | 4101.5 | 5785.5 | 7840.4 | 9499 | 10253 | 9873.4 | 9773.2 | 9555 | 9220.7 |

Table 4.8: Results from computations using $P_1$. The values shown are the average calculated value, the best calculated value, the ratio between those two values and calcualtion time in milli seconds.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|
| **Avrage** | 59 | 102.8 | 168 | 231.8 | 286.9 | 361 | 394.4 | 424.8 | 430 |
| **Best** | 60 | 111 | 171 | 232 | 301 | 361 | 411 | 432 | 432 |
| **Ratio** | 0.98333 | 0.92613 | 0.98246 | 0.99914 | 0.95316 | 1 | 0.95961 | 0.98333 | 0.99537 |
| **CPU** | 4482.7 | 6377.9 | 8565 | 10623 | 11628 | 11268 | 10987 | 10325 | 10065 |

Table 4.9: Results from computations using $P_2$. The values shown are the average calculated value, the best calculated value, the ratio between those two values and calcualtion time in milli seconds.

is not straight forward but uses a remainder of a path to reach the best known objective value.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|
| **Avrage** | 58.5 | 109.8 | 166.8 | 221.5 | 271.1 | 324.5 | 376.9 | 402.9 | 415.9 |
| **Best** | 60 | 111 | 171 | 232 | 301 | 361 | 402 | 432 | 432 |
| **Ratio** | 0.975 | 0.98919 | 0.97544 | 0.95474 | 0.90066 | 0.89889 | 0.93756 | 0.93264 | 0.96273 |
| **CPU** | 4316.8 | 5967.1 | 7644.8 | 9889.2 | 10869 | 10282 | 10004 | 9755.3 | 9529.8 |

Table 4.10: Results from computations using $P_3$. The values shown are the average calculated value, the best calculated value, the ratio between those two values and calcualtion time in milli seconds.

In tables 4.8, 4.9 and 4.10 average run times for the algorithm is displayed. Of the three matrices $P_2$ appears to have the longest run time. The differances between run times can be explained by the odds of invoking a certain move. For example an *insert move* in very simple and uses few calculaltions where a *bus move* uses methods similar to a single *insert move* more than once. Therefore different moves have different run times and the three matrices give different average run times.

### 4.5.2 Results for Data Sets 3_50_$a, b$ and $c$

After looking at each result individually[11], for the data sets containing 50 points and 3 routes, the results were looked at as a whole. In Figure 4.24 these results can be viewed. They show the best known objective value, the results calculated by the algorithm and the residual ratio.

The feasible points for each $M$ and the best known objective values, calculated by hand, for data sets 3_50_$b$ and 3_50_$c$ are shown in Table 4.11.

| **M** | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|
| 3_50_$b$ | 9 | 22 | 32 | 41 | 45 | 45 | 45 | 45 | 45 |
| **Best** | 55 | 120 | 180 | 250 | 301 | 361 | 411 | 432 | 432 |
| 3_50_$c$ | 10 | 22 | 32 | 42 | 45 | 45 | 45 | 45 | 45 |
| **Best** | 55 | 111 | 181 | 240 | 300 | 360 | 411 | 432 | 432 |

Table 4.11: Shows the number of points feasible for each $M$ and the best known objective value for data sets 3_50_$b$ and 3_50_$c$.

When inspecting Figure 4.24 it is obvious that $P = P_2$ returns the best results. One can see this by comparing the graphs in the center column to the other graphs in Figure 4.24. The residual ratio, green line, is considerably lower for graphs in the center column than the graphs situated on the right and left columns of Figure 4.24. This shows that on average the individual results of the algorithm, when using $P = P_2$, are much closer to the best known objective values, calculated by hand, than results calculated using $P_1$ or $P_3$. It should be noted that the objective function is in reality a step function and not a continuous function as displayed in Figure 4.24. The figure is set up this way to demonstrate how closely the tests follow the best

[11] Individual results for data sets 3_50_$b$ and 3_50_$c$ can be viewed in appendix $D$.1.1 and $D$.1.2.

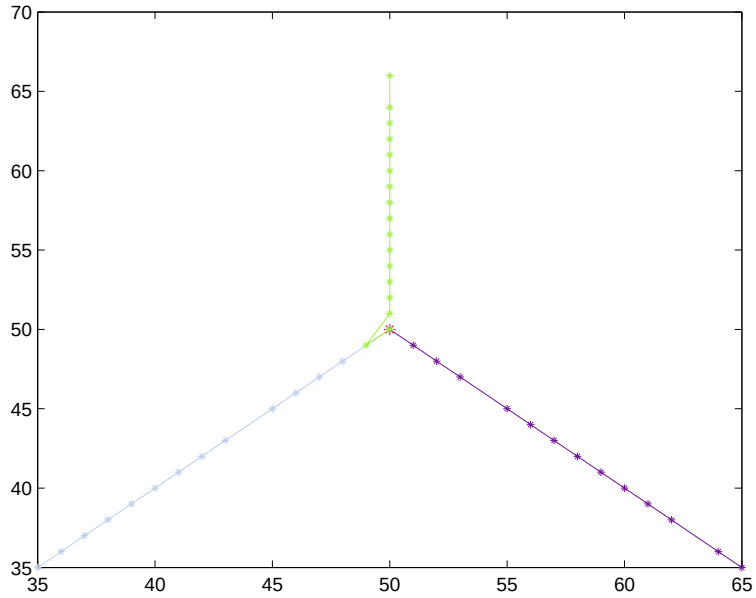



Figure 4.23: Shows results for the data set 3_50_$a$ when $M = 35$ and the an objective value of 411. Note this solution was found with $P = P_2$ in moves.java, although it took 22 runs to produce the result. Also compare with Figure 4.22 to see if all point of low profit are included.

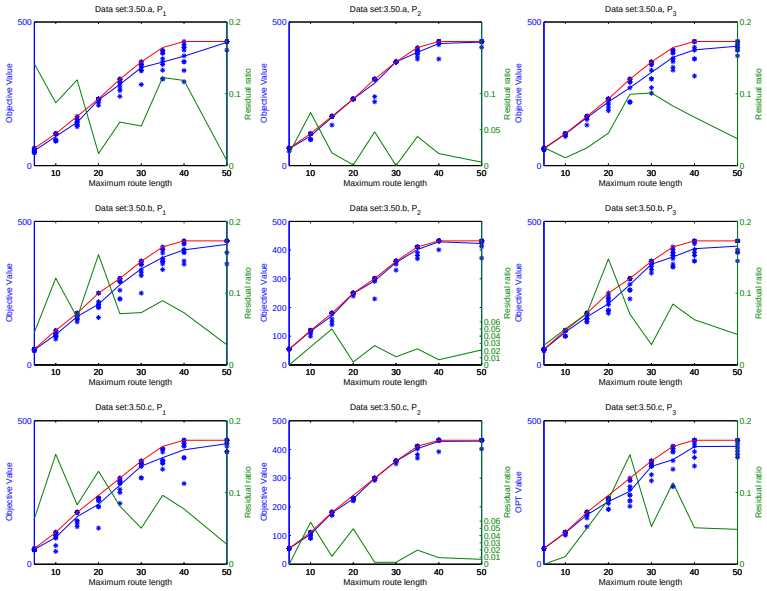



Figure 4.24: Shows the best known objective values, red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line.

known objective values.

In Figure 4.25 the run times for every test and there average function can be viewed. Note how the run times increase with $M$, reach a high point, when $M = 25$, and then decrease slightly for $M > 25$. The reason for this is that $M = 25$ allows for all points to be included, which means they are all feasible. Although all points are feasible this does not mean that all points will be included in the solution. $M = 25$ has the lowest objective value, calculated by hand, with the highest number of points at feasible points. This means all other solutions either have fewer feasible points or higher objective values, calculated by hand. Therefore $M = 25$ should have the largest number of claculations[12].

In conclusion it is apperant that $P_2$ gives the best results for data sets $3\_50\_a$, $3\_50\_b$ and $3\_50\_c$. This can be seen by comparing the residual ratios in Figure 4.24. The run time of the algorithm also is the greatest when the highest number of feasible points yields the lowest objective values. This is obtained by comparing Figure 4.25 and Table 4.11

### 4.5.3 Results for Data Sets $3\_100\_a, b$ and $c$

A graph displaying the individual results for $3\_100\_a, b$ and $c$ was constructed and can be viewed in Figure 4.26. The center column of Figure 4.26 show lower residual ratios than the graphs in the left and right column. This means that when $P_2$ is used it gives, on average, better results than when $P_1$ or $P_3$ is used. If compared with 4.24 it appears that the results

[12]For example uses UnvistedPoints.java most often.

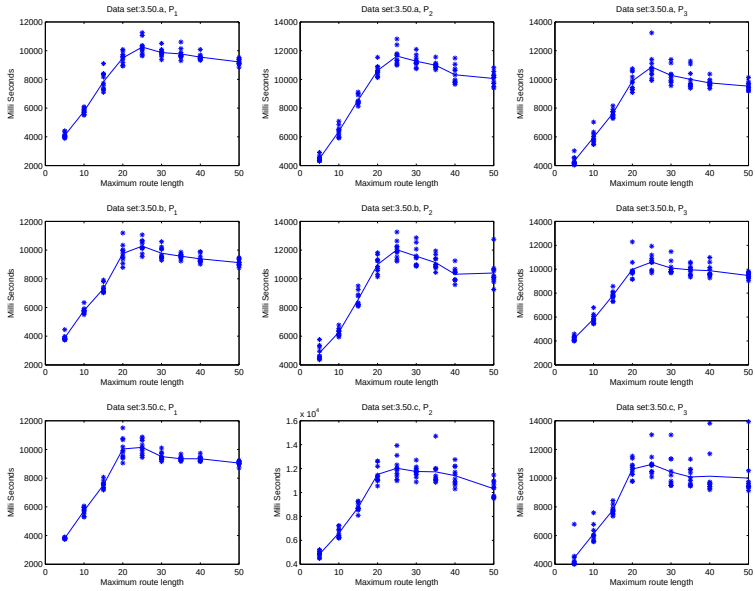



Figure 4.25: Shows how long each run of the program took, blue dots; and the average running time, blue line; with regards to maximum route length.

using 100 point give higher residual ratios on average. This means that the algorithm performs better when using smaller data sets than when using larger ones. This is most likely because the number of iterations is the same in both cases and therefore it is normal that objective function of larger data sets are more difficult to calculate. This can also be seen when inspecting the residual ratios on any single graph in Figure 4.26. Notice how the residual ratios increases as the values of $M$ gets larger.

In Table 4.12 the number of feasible points and the best known objective values, calculated by hand, are displayed. If Table 4.12 and Figure 4.27 it can be seen that the run time of the algorithm rizes till it reaches $M = 50$ and the falls slightly. This is because $M = 50$ gives the lowest objective value while having the largest set of feasible points. This is when $M = 50$ and allows all 94 points to be included in the calculation but the best known objective value is lower than with $M = 60$ or $M = 75$.

In conclusion it can be seen from Figure 4.26 that $P_2$ is the best probabilty matrix for data sets 3_100_$a, b$ and $c$. When Figure 4.26 was inspected it was observed that residual ratios rize as the values of $M$ get larger. Also for data sets 3_100_$a, b$ and $c$ the algorithm gives the highest run times when the set of feasible points is the largest while the calculating the lowest objective value.

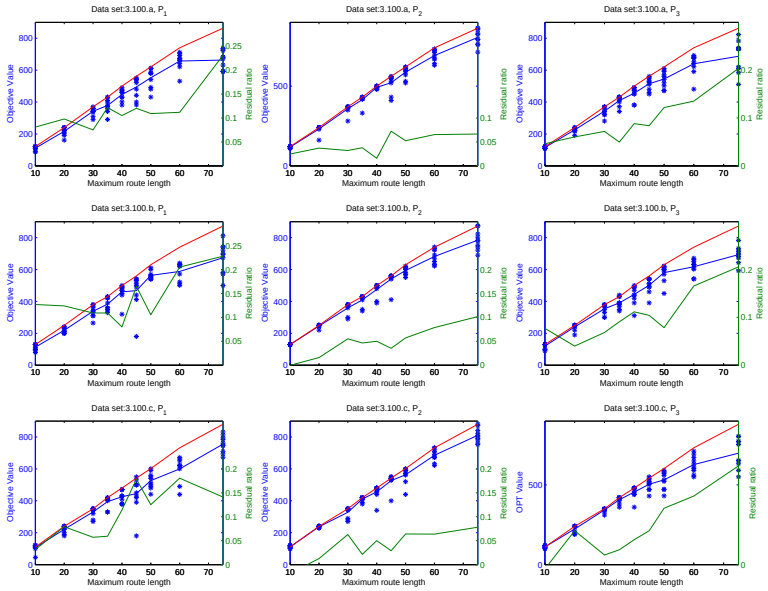



Figure 4.26: Shows the best known objective values, red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line.

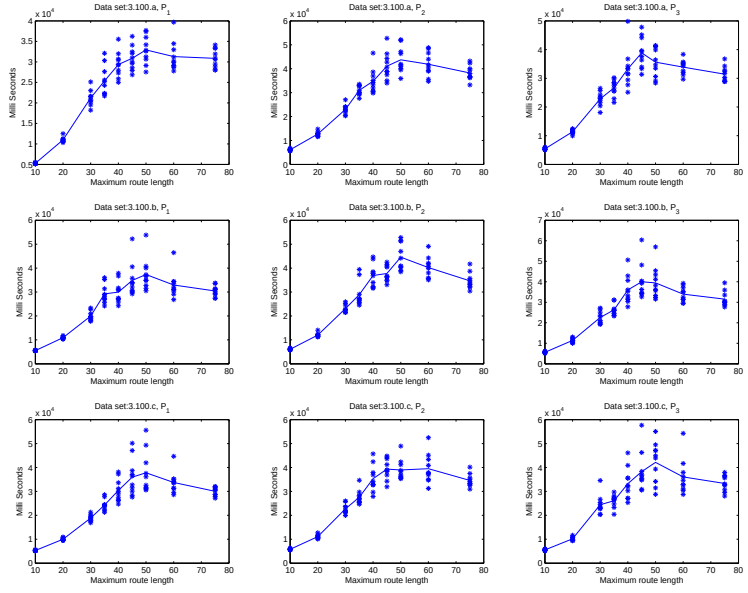



Figure 4.27: Shows how long each run of the program took, blue dots; and the average running time, blue line; with regards to maximum route length.

| **M** | 10 | 20 | 30 | 35 | 40 | 45 | 50 | 60 | 75 |
|---|---|---|---|---|---|---|---|---|---|
| v3_100_$a$ | 21 | 45 | 67 | 77 | 85 | 90 | 94 | 94 | 94 |
| **Best** | 121 | 240 | 370 | 431 | 500 | 562 | 620 | 740 | 863 |
| 3_100_$b$ | 22 | 45 | 67 | 77 | 85 | 90 | 94 | 94 | 94 |
| **Best** | 130 | 250 | 380 | 430 | 500 | 560 | 630 | 740 | 873 |
| 3_100_$c$ | 21 | 42 | 66 | 76 | 84 | 90 | 94 | 94 | 94 |
| **Best** | 110 | 240 | 350 | 420 | 480 | 542 | 602 | 732 | 880 |

Table 4.12: Shows the number of points feaslable for each $M$ and the best known objective value for data sets 3_100_$a$, 3_100_$b$ and 3_100_$c$.

### 4.5.4 Results for Data Sets 4_50_$a, b$ and $c$

As with previous data sets graphs were constructed to compare calculated results with the best known objective values and also to inspect the residual ratio. These graphs can be seen in Figure 4.28. The best performance is again achived by using $P = P_2$ and it is considerably better the that of $P_1$ and $P_3$. Results calculated with $P_2$ are displayed in the center column and they have a considerably lower residual ratio than the graphs in the left and right columns. This data set was allowed to plateau, best know objective for $M \in \{24, 28, 30, 35\}$ is 432, the combined profit of all the nodes.

Plots of the running times were also done, see figure 4.29. The sharp increase in run times results in the highest values when $M = 12$, which is the lowest objective value, calculated by hand, in concern with the number of feasible points. This is the same for all three data sets and can be seen in Table 4.13

| **M** | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 30 | 35 |
|---|---|---|---|---|---|---|---|---|---|
| 4_50_$a$ | 14 | 33 | 45 | 45 | 45 | 45 | 45 | 45 | 45 |
| **Best** | 70 | 131 | 201 | 281 | 361 | 432 | 432 | 432 | 432 |
| 4_50_$b$ | 15 | 34 | 45 | 45 | 45 | 45 | 45 | 45 | 45 |
| **Best** | 70 | 150 | 211 | 290 | 370 | 432 | 432 | 432 | 432 |
| 4_50_$c$ | 16 | 34 | 45 | 45 | 45 | 45 | 45 | 45 | 45 |
| **Best** | 80 | 160 | 231 | 291 | 372 | 432 | 432 | 432 | 432 |

Table 4.13: Shows the number of points feaslable for each $M$ and the best known objective value for data sets 4_50_$a$, 4_50_$b$ and 4_50_$c$.

In conclusion it is apperant that calculations with $P_2$ give better solutions than those using $P_1$ or $P_3$ for data sets 4_50_$a$, 4_50_$b$ and 4_50_$c$. This can be observed in Figure 4.28 by comparing the center column graphs to other graphs in the figure. Run times for the algorithm are also the highest when looking at the lowest best known objective value, calculated by hand, in concern with the largest amount of feasible points. This is observed by comparing Figure 4.29 and 4.13.

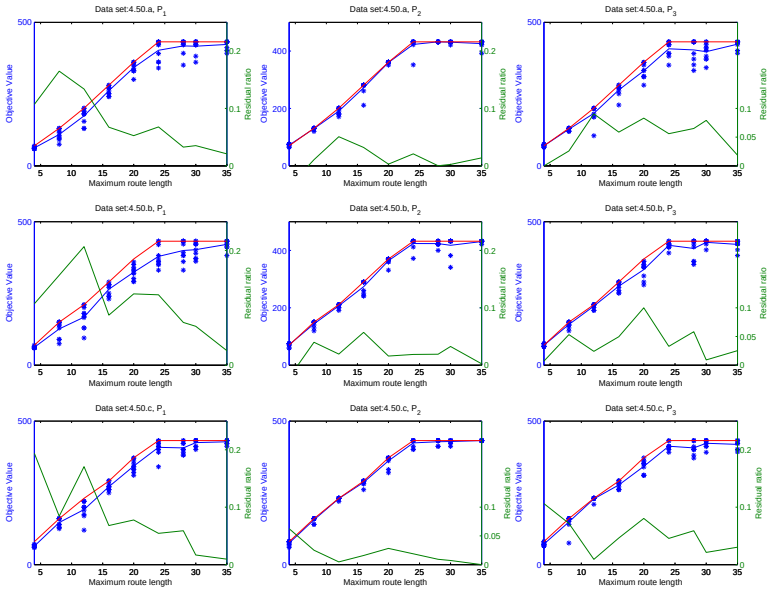



Figure 4.28: Shows the the best known objective values red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line.

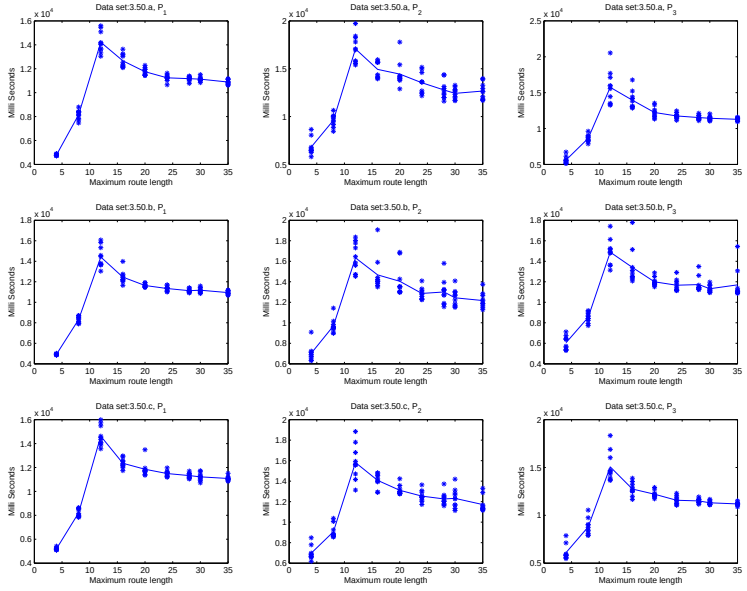



Figure 4.29: Shows how long each run of the program took, blue dots; and the average running time, blue line; with regards to maximum route length.

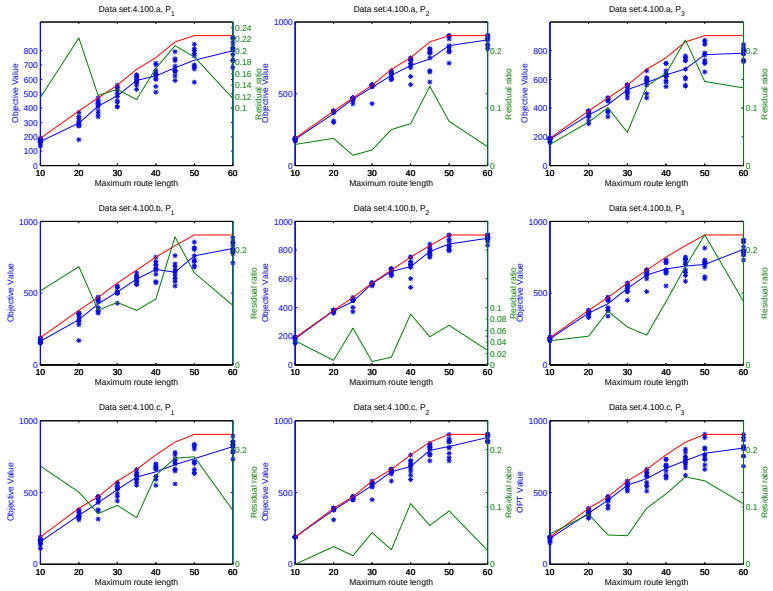



Figure 4.30: Shows the best known objective values, red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line.

### 4.5.5 Conclusion in Data Sets 4_100_$a, b$ and $c$

Graphs displaying the results for 4_100_$a, b$ and $c$ can be found in Figure 4.30. If the three columns in Figure 4.30 are compared it can be seen that the graphs in center column give the lowest residual ratio. These results, the ones displayed in the center column of Figure 4.30, were calculated using $P_2$ whilst the the other results were found using $P_1$ or $P_3$. If compared with results in Figure 4.28 it is apparent that the graphs displayed in Figure 4.28 give lower residual ratios. Although when Figure 4.30 is inspected the residual ratios do not increase, on average, as $M$ gets larger. This does though occur in some graphs in Figure 4.30, observe the right column.

| **M** | 10 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 60 |
|---|---|---|---|---|---|---|---|---|---|
| 4_100_$a$ | 34 | 73 | 95 | 95 | 95 | 95 | 95 | 95 | 95 |
| **Best** | 190 | 380 | 471 | 560 | 670 | 752 | 861 | 905 | 905 |
| 4_100_$b$ | 34 | 72 | 95 | 95 | 95 | 95 | 95 | 95 | 95 |
| **Best** | 190 | 380 | 470 | 570 | 660 | 751 | 831 | 905 | 905 |
| 4_100_$c$ | 34 | 73 | 95 | 95 | 95 | 95 | 95 | 95 | 95 |
| **Best** | 190 | 390 | 470 | 580 | 661 | 762 | 851 | 905 | 905 |

Table 4.14: Shows the number of points feaslable for each $M$ and the best known objective value for data sets 4_100_$a$, 4_100_$b$ and 4_100_$c$.

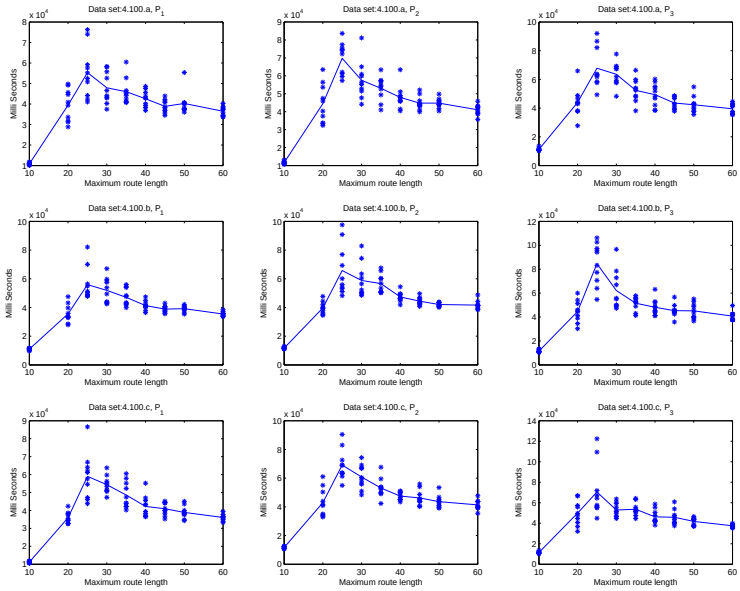



Figure 4.31: Shows how long each run of the program took, blue dots; and the average running time, blue line; with regards to maximum route length.

Table 4.14 shows the feasible points for each $M$ and the best known objective values, calculated by hand. If Table 4.14 and Figure 4.31 are compared it can be seen that the largest run times are the result of the sets with large sets of feasible points but a low objective value. This is when $M = 25$ allows for all 95 points to be included in the calculation. This results in the lowest objective value for all values $M$ giving 95 feasible points.

In conclusion from Figure 4.30 it was observed that $P_2$ was best suited for calculations with data sets $4\_100\_a, b$ and $c$. Also table 4.14 and Figure 4.31 showed that the largest run times are found when the largest set of feasible points result in the lowest best known objective value.

## 4.6 Randomly Generated Data Sets

In these test the randomly constructed data sets were used. There were 10 of these tests constructed, 5 with 50 points and 5 with 100 points. As the points are randomly distributed there was only a need to construct one profit vector for 50 points and one for 100 points. These profit vectors can be viewed in appendix $D$.3.3.

In table 4.15 the number of feasible points for a certain $M$ is displayed for randomly generated data sets.

| **M** | 10 | 20 | 40 | 50 | 70 | 80 | 100 | 130 | 160 |
|---|---|---|---|---|---|---|---|---|---|
| *50a* | 1 | 4 | 27 | 38 | 50 | 50 | 50 | 50 | 50 |
| *50b* | 3 | 4 | 21 | 39 | 50 | 50 | 50 | 50 | 50 |
| *50c* | 1 | 3 | 20 | 38 | 50 | 50 | 50 | 50 | 50 |
| *50d* | 1 | 4 | 21 | 37 | 50 | 50 | 50 | 50 | 50 |
| *50c* | 3 | 7 | 27 | 40 | 50 | 50 | 50 | 50 | 50 |
| *100a* | 2 | 11 | 51 | 78 | 100 | 100 | 100 | 100 | 100 |
| *100b* | 3 | 13 | 49 | 77 | 99 | 100 | 100 | 100 | 100 |
| *100c* | 4 | 10 | 49 | 79 | 100 | 100 | 100 | 100 | 100 |
| *100d* | 4 | 11 | 47 | 77 | 100 | 100 | 100 | 100 | 100 |
| *100c* | 2 | 12 | 49 | 75 | 100 | 100 | 100 | 100 | 100 |

Table 4.15: Show the number of feasible points for a certain $M$ in a data set of randomly generated points.

### 4.6.1 Test with data sets 50a,b,c,d and e with 450,000 iterations

In tests with non-randomly constructed data sets it was determined that $P_2$ gave better solutions than $P_3$ and $P_1$. Because $P_3$ was calculated using randomly generated data set and $P_2$ was calculated using a non-randomly constructed data set a second comparison was deemed necessary. Therefore these two probability matrices were compared again using randomly generated data sets.

The cooling schedule used in these tests was $T_0 = 15$, $r = 1 - 10^{-13}$ and $F = 2$. Also Decrease.java and the updated simulated annealing was used. In these test a large number of iterations was used or 450,000. The initial solution was the empty solution. Ten runs were performed for each possible combination for $M \in \{10, 20, 40, 50, 70, 80, 100, 130, 160\}$, $|K| \in \{3, 4, 5\}$ and $p \in \{P_2, P_3\}$.

#### Results from data sets 50a,b,c,d and e with 450,000 iterations

Plots for each data set can be viewed in the appendix. As previously the residual ratios are important. So for each data set and each value $|K| \in \{3, 4, 5\}$ the residual ratios were aggregated. From this tables 4.16 and 4.17 were constructed.

| **M** | 10 | 20 | 40 | 50 | 70 | 80 | 100 | 130 | 160 | **AVE** |
|---|---|---|---|---|---|---|---|---|---|---|
| \|K\|=3 | 0 | 0.1600 | 0.3075 | 0.7020 | 1.3472 | 1.2314 | 0.9670 | 0.2031 | 0.0006 | 0.5465 |
| \|K\|=4 | 0 | 0.1550 | 0.4850 | 0.6800 | 1.3972 | 0.9501 | 0.5440 | 0.5400 | 0.4596 | 0.5790 |
| \|K\|=5 | 0 | 0.1700 | 0.7550 | 0.8440 | 1.2044 | 1.0225 | 0.6990 | 1.0501 | 0.8438 | 0.7321 |

Table 4.16: Shows values of residual ratios calculated with $P_2$ for each $M$ and $|K|$. Individual tables for each dat ser (50a,b,c,d and e) wher ussed to construct this table.

The values form Table 4.18 are plotted in Figure 4.32. As values from Table 4.17 are withdrawn from values in 4.16 Table 4.18 is constructed. It is apparent that if more values, in

| **M** | 10 | 20 | 40 | 50 | 70 | 80 | 100 | 130 | 160 | **AVE** |
|---|---|---|---|---|---|---|---|---|---|---|
| \|K\|=3 | 0 | 0.1600 | 0.6775 | 0.5240 | 1.6657 | 1.3713 | 1.1860 | 0.3609 | 0.0106 | 0.6618 |
| \|K\|=4 | 0 | 0.1350 | 1.2800 | 0.7540 | 1.3043 | 0.9237 | 0.5920 | 0.5500 | 0.4033 | 0.6603 |
| \|K\|=5 | 0 | 0.1300 | 1.5000 | 1.8300 | 1.1043 | 0.7373 | 0.7200 | 0.9577 | 0.7126 | 0.8547 |

Table 4.17: Shows values of residual ratios calculated with $P_3$ for each $M$ and $|K|$. Individual tables for each dat ser (50a,b,c,d and e) wher ussed to construct this table.

| **M** | 10 | 20 | 40 | 50 | 70 | 80 | 100 | 130 | 160 | **AVE** |
|---|---|---|---|---|---|---|---|---|---|---|
| \|K\|=3 | 0 | 0 | -0.3700 | 0.1780 | -0.3185 | -0.1399 | -0.2190 | -0.1578 | -0.0100 | -0.1152 |
| \|K\|=4 | 0 | 0.0200 | -0.7950 | -0.0740 | 0.0929 | 0.0264 | -0.0480 | -0.0100 | 0.0563 | -0.0813 |
| \|K\|=5 | 0 | 0.0400 | -0.7450 | -0.9860 | 0.1001 | 0.2852 | -0.0210 | 0.0924 | 0.1312 | -0.1226 |

Table 4.18: Is constructed from tables 4.16 and 4.17 by withdrawing values in the latter table from values in the former table.

Table 4.18, are positive, then $P_3$ is better else $P_2$ gives better objective values. If Figure 4.32 is inspected it is apparent that more values are less than zero, negative. Therefore one can assume that $P_2$ gives better objective values.

The reason this was done instead of just comparing plots is that results were to similar in comparison, using plots like those constructed for non-randomly generated data sets. In conclusion it is apparent that $P_2$ gives better results than $P_3$ for randomly generated data sets.

### 4.6.2 Comparing Probability matrices $P_2$ and $P_2^*$

After determining that the probability matrix $P_2$ gives better objective values than $P_3$ and $P_1$ it is ideal to see if $P_2^*$ gives better or equally as good results. Tests were performed on data sets 100a,b,c,d and e using a $|K| = 3$. For each $M \in \{10, 20, 40, 50, 70, 80, 100, 130, 160\}$ ten trials were run. The cooling schedule used was $T_0 = 15$, $r = 1 - 10^{-13}$ and $F = 2$. The tests used the updated simulated annealing and Decrease.java.

#### Results in Comparing Probability matrices $P_2$ and $P_2^*$

In Figure 4.33 the results from the tests can be observed. From that figure one can see that there is little differance between using $P_2$, left column, and $P_2^*$ right column.

To compare further a residual plot was constructed to inspect the differance between the two matrices further. This plot can be seen in Figure 4.34. The residual ratio results, calculated with $P_2^*$, were withdrawn from the residual ratios calculated with $P_2$. In Figure 4.34 more of the results are negative and thereby the residual results for $P_2^*$ were greater than those calculated with $P_2$.

In conclusion we have seen from Figure 4.33 that the differance between results calculated with $P_2$ and $P_2^*$ are not great. Furhter inspection, seen in Figure 4.34, showed that $P_2$ tended

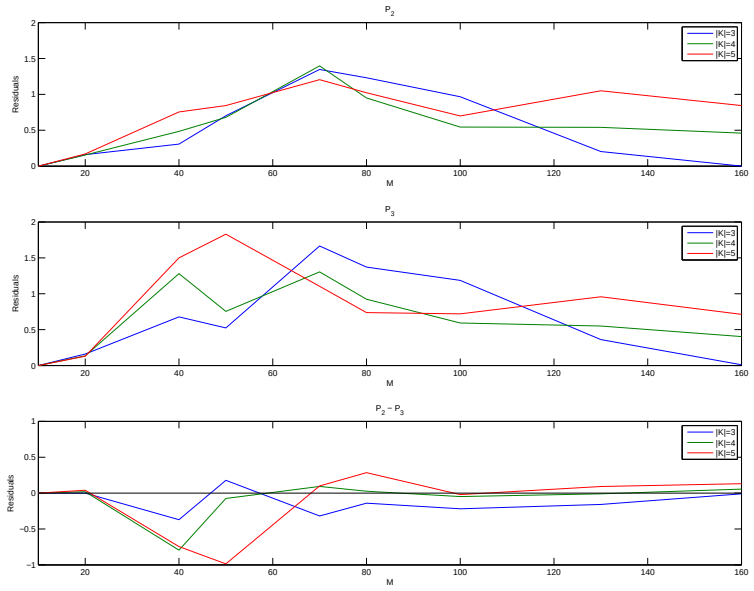



Figure 4.32: Show the residual sums for $P_2$, $P_3$ and $P_2 - P_3$

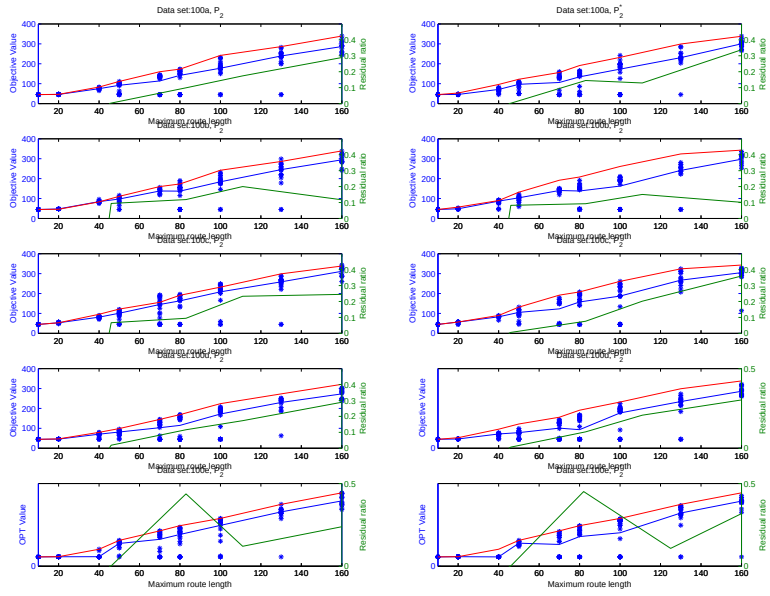



Figure 4.33: Shows the best known objective values, red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line.

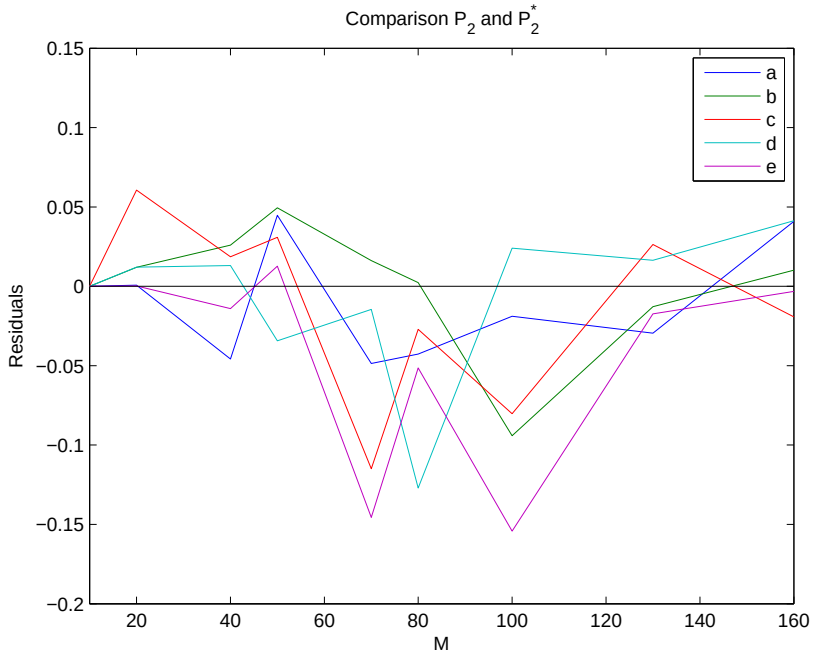



Figure 4.34: Show the result when residual sums for $P_2^*$ are removed from the residual sums of $P_2$.

to give better results on average.

## 4.7 Comparison to GAMS

A 20 point subset data set was constructed from data set 50a. This was done in effort to compare solutions from the algorithm to solutions from a differrent program. Thit other program was written in GAMS.
The model used in GAMS is the linear model presented in the section on the model.

The cooling schedule used in these tests was $T_0 = 15$ and $r = 1 - 10^{-13}$ and stopping criteria $F = 2$. Each run inspected 9 possible values for maximum route length, $M \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$, and there were 20 trials for each $M$. The initial guess is the empty solution where no routes are active. The maximum amount of profit available from the nodes was 110.
These tests also used Decrease.java along with the improved simulated annealing. In the objective function $\alpha = 1$ and $\beta = 15$.

### 4.7.1 Results Comparison to GAMS

In Table 4.19 all solutions from GAMS and the simulated annealing algortihm are compared. The solutions presented by gams had the best calculated objective, an ubber bound and the gap between the two. The best calculated objectiv will be called the lower bound as GAMS

has proven that the objective function is at least this value. If Table 4.19 is inspected it can be seen that the best values presented by the simulated annealing algorithm are all in between the lower bound and the upper bound, calculated by GAMS. Although in all cases the upper bound proposed by GAMS is:

$$\sum_{V_M} \phi_i + |K|\beta \tag{4.7.1}$$

This is an upper bound that was shown in the section on upper bounds.

| **M** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | $\|K\|$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **Best** | 45 | 49 | 49 | 59 | 69 | 76 | 95 | 107 | 110 | 3 |
| **AVE** | 45 | 48.2 | 49 | 50.45 | 64.55 | 73.3 | 89.05 | 99.2 | 107.45 | 3 |
| **LB** | 45 | 49 | 49 | 59 | 59 | 61 | 92 | 78 | 92 | 3 |
| **GAP** | 9 | 20 | 20 | 45 | 69 | 65 | 63 | 68 | 63 | 3 |
| **UB** | 54 | 69 | 69 | 99 | 123 | 139 | 155 | 155 | 155 | 3 |
| **Best** | 60 | 64 | 64 | 74 | 84 | 92 | 109 | 119 | 125 | 4 |
| **AVE** | 60 | 63.6 | 63.8 | 65.9 | 80.75 | 86.55 | 101.65 | 111.3 | 113.35 | 4 |
| **LB** | 60 | 64 | 64 | 74 | 74 | 76 | 107 | 102 | 107 | 4 |
| **GAP** | 9 | 20 | 20 | 40 | 64 | 65 | 63 | 68 | 63 | 4 |
| **UB** | 69 | 84 | 84 | 114 | 138 | 141 | 170 | 170 | 170 | 4 |
| **Best** | 75 | 79 | 79 | 89 | 99 | 106 | 124 | 129 | 138 | 5 |
| **AVE** | 75 | 79 | 78.8 | 80.45 | 95.85 | 101.15 | 118.85 | 122.85 | 125.85 | 5 |
| **LB** | 75 | 79 | 79 | 89 | 89 | 91 | 122 | 117 | 122 | 5 |
| **GAP** | 9 | 20 | 20 | 40 | 64 | 78 | 63 | 77 | 63 | 5 |
| **UB** | 84 | 99 | 99 | 129 | 153 | 156 | 185 | 185 | 185 | 5 |

Table 4.19: Shows the average results from the simulated annealing algorithm and its maximum calculated values. This is compared with values calculated by GAMS, the upper limit proposed by GAMS and the gap between the two.

A plot of the results can be viewed in Figure 4.35. The greatest variation found between average value and the best known objective value is under 20%. This is not perfect but acceptable.

The comparison between GAMS and the simulated annealing algorithm show that the best value calculated with simulated annealing is always closer to the best known objective, than the value calculated with GAMS. Also simulated annealing much faster as a single calculation in GAMS more then a day finish but took only seconds using the simulated annealing algorithm.

The GAMS code performed well for $M \leq 50$, this is most likely because when $M = 50$ there are only 12 points within the maximum route length, this increases to 17 for $M = 60$ and 20 for $M \geq 70$. This shows that with increasing number of points the GAMS program has more difficulty calculating the objective value, which was expected. There fore the GAMS does gives to low values when calculating for $M \geq 60$.
The simulated annealing algorithm also has worse average values for high values of $M$. This

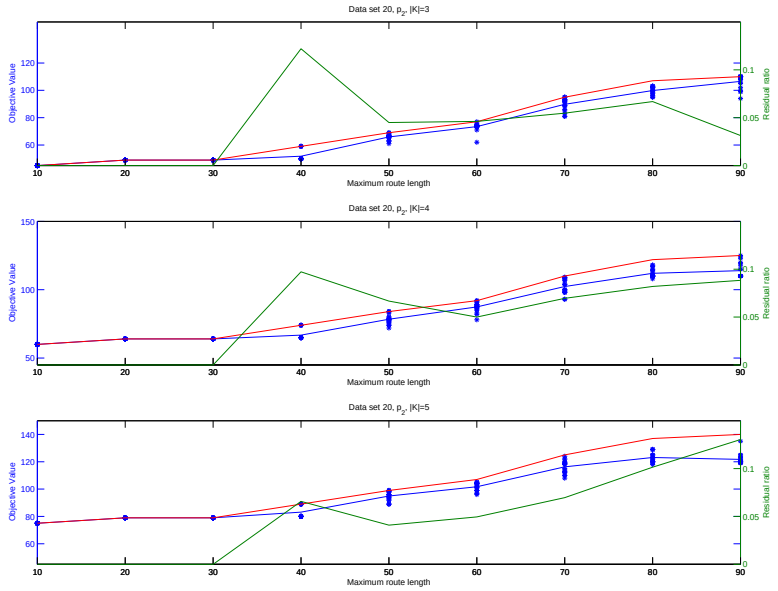



Figure 4.35: Shows the best known objective values, red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line. These are the results when using simulated annealing for both time and profit, used Decrease.java and a new cooling schedule.

is also due to larger number of available nodes and can be seen in figure 4.35.

From this, Table 4.19 and Figure 4.35, one can conclude that the simulated annealing algorithm is performing accaptably weel in comparison with GAMS and using data set 20.

### 4.7.2 Results Comparison to Decrease.java with New Cooling Schedule

A comparision between using Decrease.java and not was done. Also when Decrease.java was not used an older cooling schedule was still in use. There $T_0 = 3000$, $r = 0.999$ and $F = 0$. Results from using Decrease.java, and the new cooling schedule, can be seen in Figure 4.35 while results from not using it are seen in Figure 4.36. Note that both trials used an updated version of the simulated annealing algorithm. When these two figures are compared it is apparent that results are better with Decrease.java and the new cooling schedule. A table containing results from the older version can be seen in appendix $D$.4.

Also run times were compared to see which method was faster. These results are seen in tables 4.20 and 4.21.

As can be seen in tables 4.21 and 4.20 run times for low values of $M$ are lower when Decrease.java is used. In that case, using the new version of the algorithm, the run time is greates for $M = 60$ were all points are feasible but the objective value is the lowest with comparison

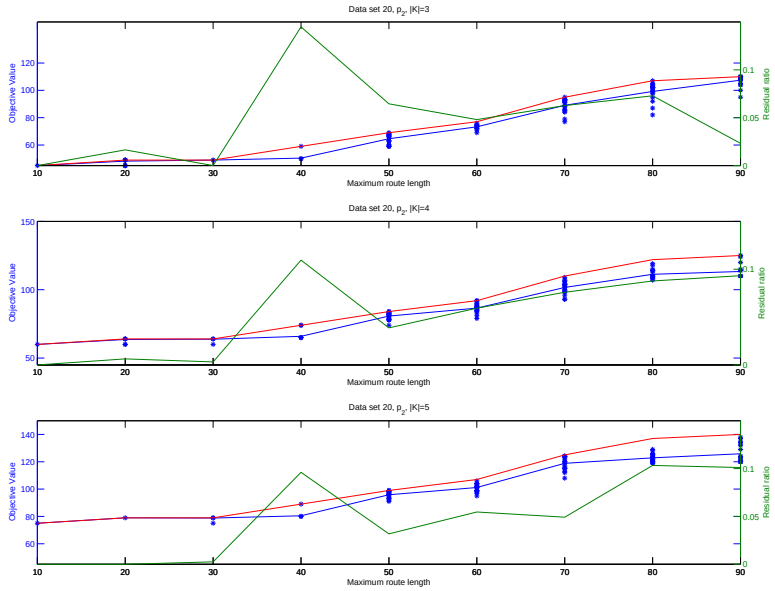



Figure 4.36: Shows the best known objective values, red line; the different results and there average value, blue dots and blue line; also the residual ratio is plotted, green line. These are the results when using simulated annealing for both time and profit.

| **M** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | $\|K\|$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **mean** | 3208 | 3520 | 3588 | 4711 | 5418 | 6194 | 5484 | 5448 | 4991 | 3 |
| **max** | 4068 | 6156 | 6257 | 6459 | 8222 | 9479 | 8899 | 8541 | 6304 | 3 |
| **min** | 3009 | 3227 | 3208 | 4425 | 4406 | 5091 | 5036 | 5035 | 4505 | 3 |
| **mean** | 7433 | 8147 | 8134 | 11384 | 13195 | 17023 | 14649 | 12603 | 11389 | 4 |
| **max** | 10362 | 11848 | 10585 | 16854 | 18304 | 27887 | 21553 | 17612 | 16120 | 4 |
| **min** | 5172 | 5797 | 5572 | 8168 | 9264 | 11974 | 9775 | 8142 | 7686 | 4 |
| **mean** | 6619 | 7148 | 7275 | 10591 | 12669 | 16453 | 13278 | 10898 | 9772 | 5 |
| **max** | 9404 | 9787 | 9855 | 14740 | 18074 | 26861 | 20284 | 15144 | 14207 | 5 |
| **min** | 5156 | 5828 | 5611 | 8394 | 10073 | 13374 | 8211 | 7198 | 7657 | 5 |

Table 4.20: Shows run times for tests using the new cooling schedule and using Decrease.java, results are in milli seconds.

to other sets with the same number of points. For the older version of the algorithm high run times are recorded when $M \in \{20, 30\}$. In that case there is a large number of infeasible solutions proposed by the algorithm, Decrease.java removes a majority of these infeasible solutions. Average run times in both cases were compared and no conclusive result could be reached. The old version performed better on average of 0.3 seconds. In conclusion the new version gave better solutions on similar run times which are good results.

| **M** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | $\|K\|$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **mean** | 7779 | 10278 | 19730 | 13120 | 7464 | 6413 | 5186 | 5050 | 4733 | 3 |
| **max** | 22591 | 23881 | 29952 | 25254 | 15219 | 16555 | 7544 | 5525 | 5398 | 3 |
| **min** | 6873 | 6858 | 10446 | 7132 | 5002 | 4998 | 4873 | 4849 | 4239 | 3 |
| **mean** | 8196 | 10742 | 10898 | 12113 | 9301 | 7946 | 6833 | 5664 | 5107 | 4 |
| **max** | 8674 | 15341 | 12453 | 21734 | 12262 | 10533 | 9089 | 7106 | 6739 | 4 |
| **min** | 7950 | 7812 | 7791 | 8131 | 8216 | 5492 | 5436 | 4854 | 4318 | 4 |
| **mean** | 8870 | 11683 | 12274 | 12721 | 9827 | 9008 | 6635 | 5695 | 5198 | 5 |
| **max** | 9233 | 13504 | 13371 | 13471 | 12987 | 12287 | 9384 | 8349 | 8483 | 5 |
| **min** | 8719 | 8793 | 8765 | 8763 | 8833 | 7861 | 4756 | 4755 | 4731 | 5 |

Table 4.21: Shows run times for tests using the older version of the algorithm, results are in milli seconds.

## 4.8 Obtained Data Sets

These previously conducted tests were used in [13]. There were 3 data sets tested each in different size. The sets sizes are $|V| = 102$, $|V| = 32$ and $|V| = 33$. There names are respectively data set 102, 32 and 33. To be comparable with the problem presented in [13] $\beta$ was set to zero. This means that the only contributing factor in the objective function is $\sum_{i \in V} p_i y_i$, as $\alpha = 1$. Now the number of possible routes was $|K| \in \{2, 3, 4\}$. The cooling schedule was set to $T_0 = 15$, $r = 1 - 10^{-13}$ and $F = 2$. The number of iterations was 50,000 and the probability matrix used was $P_2$. Each test with all possible combination was done 10 times. The maximum route lengths differed for each test.

### 4.8.1 Results for Data Set 32

The results for the comparison between the tabu search algorithm presented in [13] and the simulated annealing algorithm presented in this thesis can be seen in table 4.22. The tabue search algorithm performs better in most cases, although the simulated annealing performs equally well in a few cases and better in one ($|K| = 3$ and $M = 13.3$). In the one case where the simulated annealing algorithm performs better it finds the best known objective value. It is stated in [13] that the best known objective value for that paticular case, when $M = 13.3$ and $|K| = 3$, is 75. Overall the simulated annealing algorithm performs sufficently well compared to the tabu search.

The run times of the two methods are also dispayed in Table 4.22, these time are mesured in seconds. The computer used in the tabu search experiment was a DEC Alpha XP1000 Computer and the one used to calculated the simulated annealing was a Dell Inspiron 5150 (Pentium 4). When the two methods are compared it is obvious that the tabu search is much faster than the simulated annealing, somtimes faster by as much as 15 seconds. The values shown are both maximum calculations time recored in there trials.

In conslusion one can see, by inspecting Table 4.22 that the tabu search algortihm returns better soltutions faster than the simulated annealing algorithm, when daeiling with the 32 point data set. Although the simulated annealing algorithm does not return as good solutions,

| | | S.A. | | | | Tabu | |
|---|---|---|---|---|---|---|---|
| $\|K\|$ | $M$ | Average | Max | Min | CPU | Max | CPU |
| 4 | 18.8 | 151 | 165 | 135 | 13.011 | 175 | 1.5 |
| 4 | 18.2 | 145.62 | 155 | 130 | 12.708 | 165 | 1.3 |
| 4 | 12.5 | 72.75 | 75 | 70 | 11.964 | 75 | 0.8 |
| 3 | 25 | 188.75 | 220 | 135 | 11.737 | 220 | 1.5 |
| 3 | 24.3 | 177.88 | 195 | 140 | 11.395 | 205 | 2.6 |
| 3 | 21.7 | 147.62 | 165 | 125 | 10.208 | 170 | 1.4 |
| 3 | 13.3 | 73.0 | 75 | 70 | 11.712 | 70 | 0.8 |
| 2 | 23 | 122.25 | 135 | 95 | 15.860 | 135 | 1.3 |

Table 4.22: Show the comparison between a tabu search algorithm persented in [13] and the simulated annealing algorithm persented in this thesis.

as the tabu search, it gives resonalby good results in some cases even finding the best known objective value.

### 4.8.2 Results for Data Set 33

In Table 4.23 a comparison between a tabu search algorithm, persented in [13], and the simulated annealing used in the thesis is displayed. The tabu search performs better on average. The simulated annealing in some cases give equally good values as the tabu search but never better values. Though the tabu search performs better overall the simulated annealing tends to find objective values close to the best known objective values, persented by the tabu search algorithm.

The run times, displayed in table in Table 4.23, are compared it is apparent that tabu search is much faster than the simulated annealing algorithm. The computer used in the tabu search experiment was a DEC Alpha XP1000 Computer and the one used to calculated the simulated annealing was a Dell Inspiron 5150 (Pentium 4). The differance between runtimes is considarble with tabu search out performing the simulated annealing algorithm by as much as 12 seconds. In both cases the run time are maximum numbers recorded over a few trials.

Overall tabu search out performs simulated annealing both in concern to the objective values and run time, seen in Table 4.23, when compared with data set 33. Although simulated annealing does return good objective values but not always the best known objectives.

### 4.8.3 Results for Data Set 102

Comparison with the 102 point data set can be viewed in Table 4.24. In most instnces the tabu search algorithm returns better objective values the the simulated annealing algorithm. There are also cases where the two algorithms return the same best objective values. In one case,$M = 93.3$ and $|K| = 3$, simulated annealing returned a better objective than the tabu search algorithm. In this case the best known objective values, according to [13], is 813.

The run times are also compared in Table 4.24. Values displayed are the maximum recorded

| | | S.A. | | | | Tabu | |
|---|---|---|---|---|---|---|---|
| $\|K\|$ | $M$ | Average | Max | Min | CPU | Max | CPU |
| 4 | 22.5 | 481.5 | 520 | 400 | 12.233 | 560 | 0.7 |
| 4 | 15 | 259.25 | 280 | 220 | 11.388 | 310 | 0.8 |
| 4 | 10 | 190 | 190 | 190 | 10.008 | 190 | 0.6 |
| 3 | 36.7 | 686.25 | 720 | 620 | 10.840 | 750 | 3.3 |
| 3 | 31.7 | 610.75 | 650 | 540 | 11.818 | 680 | 3.1 |
| 3 | 30 | 574 | 620 | 470 | 10.447 | 640 | 2.1 |
| 3 | 28.3 | 534.75 | 570 | 420 | 11.562 | 590 | 2.0 |
| 3 | 25 | 471.75 | 500 | 430 | 14.083 | 510 | 2.0 |
| 2 | 47.5 | 697.5 | 740 | 630 | 8.907 | 760 | 5.4 |
| 2 | 42.5 | 619.25 | 660 | 540 | 9.775 | 690 | 6.6 |
| 2 | 30 | 425.75 | 490 | 290 | 10.973 | 490 | 1.5 |
| 2 | 27.5 | 386.75 | 430 | 280 | 9.512 | 460 | 3.8 |
| 2 | 25 | 360 | 390 | 270 | 11.092 | 410 | 3.1 |
| 2 | 20 | 261.75 | 290 | 180 | 10.296 | 290 | 1.2 |
| 2 | 17.5 | 212.5 | 250 | 170 | 9.132 | 250 | 0.8 |
| 2 | 12.5 | 176 | 180 | 110 | 11.354 | 180 | 1.2 |

Table 4.23: Show the comparison between a tabu search algorithm persented in [13] and the simulated annealing algorithm persented in this thesis.

run times. The computer used in the tabu search experiment was a DEC Alpha XP1000 Computer and the one used to calculated the simulated annealing was a Dell Inspiron 5150 (Pentium 4). In most cases the simulated annealing algorithm uses shorter runtimes but for low values of $M$ the tabu search is quicker.

In conclusion the simulated annealing algorithm returns good results but not always the best possible and calculates them in short times compared to other methods, when dealing with large data sets.

## 4.9 Distance Constraint

When dealing with routes one does not want the bus to drive a short distance and before stopping again. Two nodes close to one another share much of the same profit. Therefore a small change was implemented to one of the java classes, UnvisitedPoints.java. This ensured that the bus had to drive either for some time or a certain distance before stopping again, whether it was distance or time depends on the input. The new class UnvisitedPoints2.java made it impossible for any route to stop within a certain radius $a$, from an already picked node. Test that were conducted with UnvistedPoints2.java also used Decrease.java and the updated version of simulated annealing. The data sets tested were 50a,b,c,d and e. Each test looks at nine possible maximum route lengths $M \in \{10, 20, 40, 50, 70, 80, 100, 130, 160\}$ and had a maximum of three vehichles, $|K| = 3$. Values of the radius were $a \in \{1, 2, 3, 4, 5, 10\}$. All possible combinations of data sets, maximum route lengths and radiuses were tested 10 times and each test used two initial guesses the second being the solution from the first test.

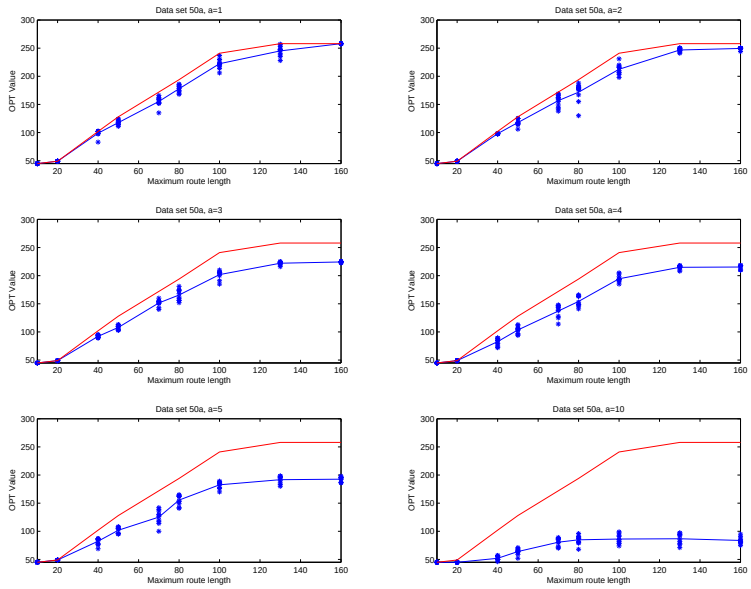



Figure 4.37: Shows the best known objective values, red line; and the different results and there average value, blue dots and blue line.

The cooling schedule used in these tests was $T_0 = 15$, $r = 1 - 10^{-13}$ and $F = 2$. Number of iterations was 50,000.

### 4.9.1 Results for the Distance Constraint

All data sets showed a decrease in objective function as the radius $a$ increased. This can be seen in the table blow and in Figure 4.37. Similar figures for data sets 50b,c,d and e were constructed and are viewable in appendix $D$.4.1.

The Table 4.25 and Figure 4.37 shows that as $a$ increases the average objective values decrease. For example the best known objective value for $M = 20$ is 49 for all values of $a \in \{1, 2, 3, 4, 5\}$ but when $a = 10$ the objective value decreases to 45. For other values of $M$ the decrease is much more obvious.

All points in $V$ for data set 50a are shown in Figure 4.38. When $a = 3$ and $M$=160 the routes chosen can be seen in Figure 4.39, routes constructed for the same $M$ and $a = 10$ is shown in Figure 4.40. Other similar figures for data set 50a can be seen in appendix $D$.4.1.

To compare between a route with $a = 0$ and a route with $a = 5$. This can be seen in figures 4.41 and 4.42. In Figure 4.41 blue points reprsent nodes not chosen. When compared to 4.42 one can determin nodes that were left out becuase they are to close to there neighbor.
These results confirm that the second version of UnvisitedPoints.java works and returns solu-

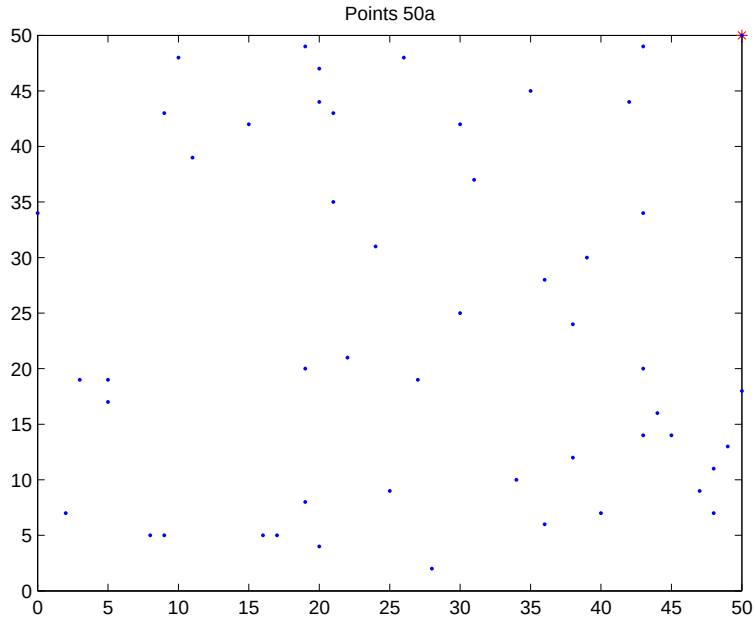



Figure 4.38: Shows all points in data set 50a.

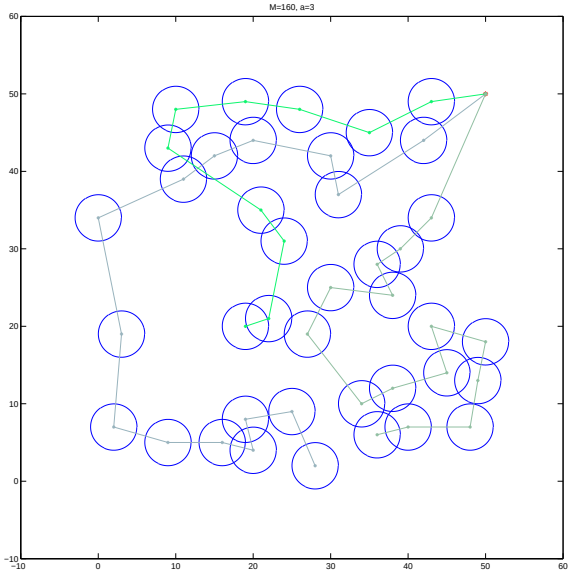



Figure 4.39: Shows the routes constructed when $a = 3$ and $M = 160$. The circles are the area where that must be travelled before another pick up point is chosen.

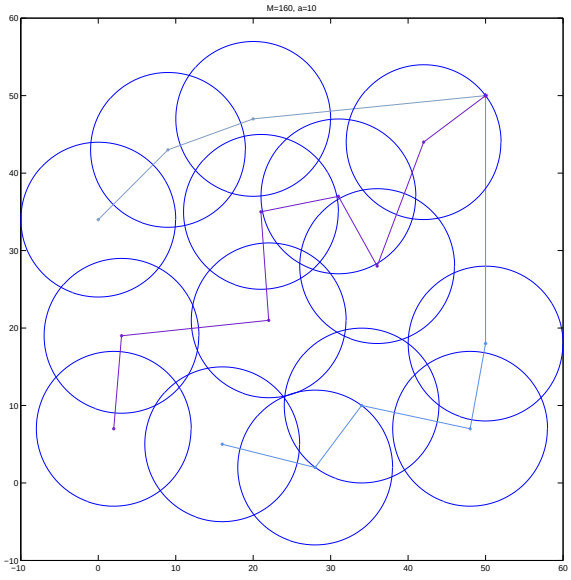



Figure 4.40: Shows the routes constructed when $a = 3$ and $M = 160$. The circles are the area where that must be traveld before another pick up point is chosen.

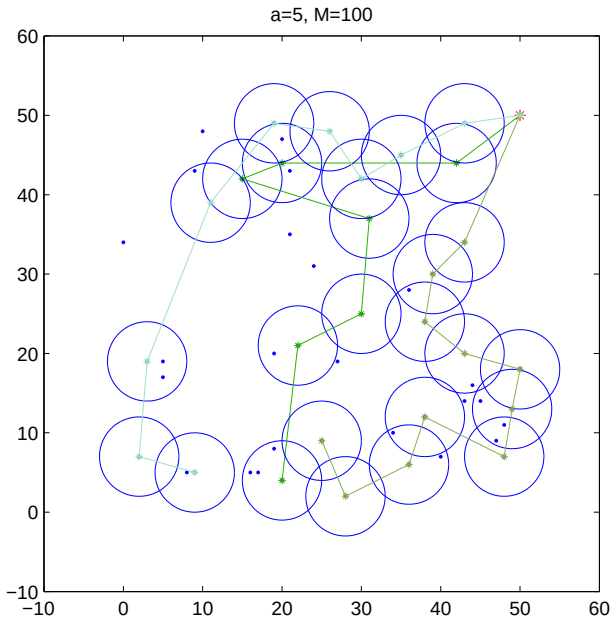



Figure 4.41: Shows the routes constructed when $a = 5$ and $M = 100$. The circles are the area where that must be travelled before another pick up point is chosen.

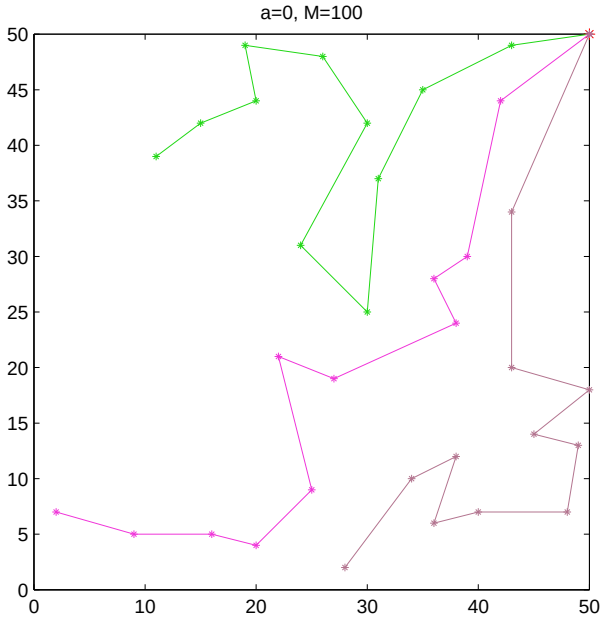



Figure 4.42: Shows the routes constructed when $a = 0$ and $M = 100$. The circles are the area where that must be travelled before another pick up point is chosen.

tions where a bus is prohibited from visiting any points within radius $a$ of a chosen node.

| | | S.A. | | | | Tabu | |
|---|---|---|---|---|---|---|---|
| $\|K\|$ | $M$ | Average | Max | Min | CPU | Max | CPU |
| 4 | 100 | 876.3 | 972 | 672 | 42.110 | 1067 | 86.6 |
| 4 | 95 | 859.8 | 1008 | 768 | 48.770 | 1019 | 84.3 |
| 4 | 90 | 813.9 | 906 | 672 | 47.007 | 966 | 101.0 |
| 4 | 85 | 729.9 | 810 | 552 | 79.156 | 905 | 95.2 |
| 4 | 80 | 681.6 | 756 | 558 | 47.383 | 832 | 82.0 |
| 4 | 75 | 660.3 | 738 | 540 | 47.741 | 776 | 71.3 |
| 4 | 70 | 592.2 | 696 | 516 | 51.924 | 726 | 54.4 |
| 4 | 65 | 543.3 | 600 | 462 | 53.606 | 643 | 68.8 |
| 4 | 60 | 483.6 | 528 | 432 | 56.333 | 576 | 31.8 |
| 4 | 55 | 418.8 | 456 | 342 | 64.727 | 503 | 44.9 |
| 4 | 50 | 382.2 | 414 | 336 | 69.247 | 462 | 23.6 |
| 4 | 45 | 346.8 | 359 | 336 | 83.772 | 359 | 20.6 |
| 3 | 133.3 | 911.7 | 1032 | 774 | 30.927 | 1098 | 143.2 |
| 3 | 126.7 | 1895.5 | 972 | 816 | 32.517 | 1061 | 99.8 |
| 3 | 120 | 863.4 | 942 | 702 | 33.138 | 1011 | 93.6 |
| 3 | 113.3 | 790.8 | 900 | 612 | 32.271 | 966 | 98.8 |
| 3 | 106.7 | 781.5 | 858 | 714 | 32.503 | 922 | 74.0 |
| 3 | 100 | 723.6 | 840 | 606 | 34.229 | 874 | 102.8 |
| 3 | 93.3 | 677.4 | 792 | 420 | 33.218 | 789 | 126.5 |
| 3 | 86.7 | 608.1 | 738 | 486 | 34.962 | 756 | 121.2 |
| 3 | 80 | 543.9 | 660 | 360 | 34.602 | 681 | 69.5 |
| 3 | 73.3 | 529.2 | 632 | 414 | 40.395 | 632 | 94.4 |
| 3 | 66.7 | 445.2 | 552 | 300 | 39.312 | 563 | 107.7 |
| 3 | 60 | 394.2 | 438 | 336 | 42.797 | 481 | 36.0 |
| 3 | 53.3 | 327 | 390 | 270 | 43.955 | 416 | 34.0 |
| 3 | 46.7 | 312.6 | 330 | 282 | 56.311 | 344 | 21.0 |
| 2 | 200 | 951 | 1062 | 810 | 26.023 | 1165 | 290.6 |
| 2 | 190 | 1932.1 | 1050 | 798 | 23.908 | 1116 | 215.6 |
| 2 | 180 | 1885.6 | 972 | 780 | 23.946 | 1067 | 432.6 |
| 2 | 170 | 852.9 | 978 | 750 | 26.709 | 1017 | 239.6 |
| 2 | 160 | 826.5 | 894 | 750 | 39.186 | 987 | 272.1 |
| 2 | 150 | 769.8 | 894 | 648 | 25.499 | 914 | 202.8 |
| 2 | 140 | 695.1 | 792 | 534 | 30.222 | 864 | 224.3 |
| 2 | 130 | 630.9 | 744 | 456 | 26.995 | 817 | 174.1 |
| 2 | 120 | 579.9 | 702 | 450 | 27.055 | 767 | 217.5 |
| 2 | 110 | 579 | 642 | 450 | 30.544 | 702 | 120.1 |
| 2 | 100 | 495 | 600 | 324 | 26.256 | 638 | 118.7 |
| 2 | 90 | 420.6 | 564 | 282 | 28.611 | 578 | 84.4 |
| 2 | 80 | 417.3 | 480 | 258 | 29.395 | 521 | 52.0 |
| 2 | 70 | 337.2 | 384 | 234 | 31.648 | 459 | 74.6 |
| 2 | 60 | 298.2 | 336 | 222 | 35.746 | 382 | 42.7 |
| 2 | 50 | 263.7 | 276 | 246 | 41.383 | 290 | 37.7 |

Table 4.24: Show the comparison between a tabu search algorithm persented in [13] and the simulated annealing algorithm persented in this thesis.

| **M** | 10 | 20 | 40 | 50 | 70 | 80 | 100 | 130 | 160 |
|---|---|---|---|---|---|---|---|---|---|
| **Best** | 45 | 49 | 102 | 128 | 172 | 194 | 241 | 258 | 258 |
| $a = 1$ | 45 | 49 | 98.90 | 117.7 | 155.2 | 177.9 | 222.3 | 245.05 | 258 |
| $a = 2$ | 45 | 49 | 98 | 117.8 | 156.7 | 171.7 | 212.5 | 246.70 | 249.4 |
| $a = 3$ | 45 | 49 | 92.20 | 108 | 151.5 | 165.6 | 201.9 | 222.10 | 224.4 |
| $a = 4$ | 45 | 49 | 82.85 | 103.5 | 137.2 | 154.3 | 194.5 | 214.80 | 215.4 |
| $a = 5$ | 45 | 49 | 82.40 | 101.9 | 125.3 | 155.5 | 182.6 | 191.80 | 192.6 |
| $a = 10$ | 45 | 45 | 52 | 64.1 | 80.7 | 84.9 | 86.3 | 86.80 | 83.8 |

Table 4.25: Compares the average calculated objective values, limited by a radius $a$ and compared to the best known objective value. All this is then done for multible value of $M$.

# Chapter 5

# Conclusions

The simulated annealing algorithm has been put through a number of tests. The best known cooling schedule, $T_0 = 21$, $r = 1 - 10^{-6}$ and $F = 10^{-6}$; was calculated in section 4.2.

The best known probability matrix was calculated in 4.3 and then compared to a number of other probability matrices (see sections 4.5 and 4.6). This best probability matrix was:

$$P_2 = \begin{pmatrix} 50 & 30 & 0 & 0 & 20 & 0 \\ 0 & 10 & 10 & 0 & 30 & 50 \\ 60 & 30 & 0 & 0 & 10 & 0 \\ 0 & 10 & 10 & 60 & 20 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 & 30 \\ 0 & 0 & 70 & 30 & 0 & 0 \end{pmatrix}$$

The best node insertion method, of those proposed, was random insertion. This was shown in section 4.4.

In comparison to a program written in GAMS the simulated annealing algorithm prooved superior, see section 4.7. The algorithm returned objectives between the upper bound, calculated by GAMS, and the objective value suggested by GAMS, used as a lower bound. The gap between the upper and lower bound proposed by GAMS was always large and therefore quality of solutions could not be shown. The simulated annealing algortihm was also much faster than the GAMS program.

The simulated annealing algorithm was then compared to a tabu search algorithm used for TOP, see section 4.8. In most cases the tabu search algorithm outperformed the simulated annealing algorithm by returning better objective values. There were though a few cases there simulated annealing found better sobjectives. For small data sets the tabu search algorithm was also faster but for larger data sets simulated annealing had shorter run times. In all cases simulated annealing returned objectives close to the best known objective values.

A constraint ensuring that a bus must travel for a certain amount of time was implemented. This constraint worked a returned routes that did not violate the constraint.

The simulated annealing algorithm designed to solve the bus route problem has been constructed. In has a good cooling schedule and a good probability matrix. Furthermore a constraint forcing a certain time to pass between stops is present if needed. The algorithm returns good results but not always the best objective values when compared to other algorithms.

## 5.1 Further Work

### 5.1.1 Real World Application

Due to time constrictions and lack of easily accessible data the project was not successful in providing a good bus routes for ALCAN Iceland. In the future a travel time matrix, including all possible pick up point will have to be constructed. This matrix is estimated in size at least as $200x200$ and could possibly be larger. This matrix would be constructed using travel plans of local buses, an algorithm constructed for measurments of travel time inside Reykjavík and real world trials. After the travel time matrix has been constructed, profits have to be assigned to each possible pick up point. These profits can be influenced by factors determined by ALCAN, for excample the number of employees living close by or access to the local bus system. Finally the vector determing in the time penalty for stopping at a certain node will have to be constructed. This factor is easily estimated by assigning each node, except the source an sink, the same penalty. If considered necessary a stopping penalty dependant on the number of people picked up can be implemented.

### 5.1.2 Algorithm Improvement

As has been discussed in the report there are many things that may be improved and inspected. The upper bound using time restrictions always assumes that one bus drives to all the nodes, in many cases this is not possible. An improvement might add a nearest neighbour algorithm to determine some sort of route lengths, or travel times. This could then be used to estimate how many buses are needed to visit the nodes selected in the upper bound.

Also in Java the inheritance of variables is a bothersome. This led to the removal of the vector $y$. By doing this a double for-loop in Unvisitedpoints.java was replaced with a triple for-loop. In future versions of the algorithm inserting $y$ into the code could reduce the run time of the program, although this could be complicated as it changes much of the algorithm.

In this report the initial guess introduced into the simulated annealing algorithm was the empty set, or a previously returned solution from the algorithm. There are other methods available in choosing good initial guess, for example adaptive memory procedure discussed in [13]. These methods could in most cases decrease run time dramatically. Although in some cases, when the best objective is an empty solution, these initial guesses result in worse solutions. This can happen in some theoratical cases but is unlikly to matter in real world application. This could be countered by implementing a new move that would remove a single node, without adding a new.

The neighbourhood, moves, could also be improved. Linking InsertMove11.java, BusMove.java and InsertMove13.java would be useful. In that case InsertMove13.java and BusMove.java would use InsertMove11.java to add points to routes. Other methods such as evolutionary cluster

search discussed in [4]. Methods such as that would although lead to longer run times, simpler methods need fewer calculations.

BusMove.java could be improved be removing a node if the new route is too long, this is done until the travel time is less than $M$. This would also lead to more calculations. As proposed in [3] one could also include a function that would try to joint the two routes with the lowest profit or travel time.

## 5.2 A Learning Experiance

Doing such a large project is a great learning experience that can benefit one in future work. A much better understanding of basic methods such as simulated annealing, and its cooling schedule; computational experiments and report writing. Also understanding of complicated operations research methods such as PCTSP, VRP, OVRP and TOP was attained. New insights into organizational skills, conducting productive meetings, communicating with persons abroad, criticizing once own work and navigating through time constrictions was gained.

Einar Leif Nielsen

# Bibliography


[1] E. Balas. Price collecting traveling salesman problem. *Networks*, 19:2621–636, 1989.

[2] Walid Ben-Ameur. Computing the initial temperature of simulated annealing. *Computational Optimization and Applications*, 29(3):369–385, 2004.

[3] J. Brandao. A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564, 2004.

[4] A.A. Chaves and L.A.N. Lorena. Hybrid algorithms with detection of promising areas for the prize collecting travelling salesman problem. *Electrical Performance of Electronic Packaging, 2005. IEEE 14th Topical Meeting on*, page 6 pp., 2006.

[5] http://alcan.is/?PageID=3.

[6] http://www.tsp.gatech.edu/index.html.

[7] Chao I-Ming, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

[8] F. Maffioli M. Dell"Amico and A. Sciomachen. A lagrangian heuristic for the prize collectingtravelling salesman problem. *Annals of Operations Research*, 81:289–306, 1998.

[9] F. Maffioli M. Dell"Amico and P. Värbrand. On the prize-collecting and the asymmetric travelling salesman problem. *Int. Trans. Opl. Res*, 2:297–308, 1995.

[10] Y. Nourani and B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A (Mathematical and General)*, 31(41):8373–8385, 1998.

[11] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, page To appear, 2005.

[12] Ronald L. Rardin. *Optimization in Operations Research.* Prentice Hall, Inc, 1998.

[13] H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers and Operations Research*, 32(6):1379–1407, 2005.

[14] Michael Townsend. *Descrete Mathematics: Applied Combinatorics and Graph Theory.* The Benjamin/Cumming Publishing Company Inc., 1987.

[15] Laurence A. Wosley. *Integer Programming.* John Wiley & sons, INC, 1998.

[16] Z. Özyurt, D. Aksen, and N. Aras. Open vehicle routing problem with driver nodes and time deadlines. *Journal of the Operational Research Society*, page To appear, 2006.

[17] Z. Özyurt, D. Aksen, and N. Aras. Open vehicle routing problem with time deadlines: Solutions methods and application. *Journal of the Operational Research Society*, page To appear, 2006.

# Appendix A

# Results

| Data set | $\|K\|$ | $M$ | Average | Max | Min | Average CPU |
|---|---|---|---|---|---|---|
| 50a | 3 | 10 | 16 | 16 | 16 | 2265.5 |
| | 3 | 20 | 25.1 | 31 | 21 | 2132.4 |
| | 3 | 40 | 108.4 | 112 | 102 | 4364.7 |
| | 3 | 50 | 136.1 | 141 | 130 | 6033.6 |
| | 3 | 70 | 172.6 | 189 | 149 | 8300.9 |
| | 3 | 80 | 202.9 | 220 | 184 | 8231.3 |
| | 3 | 100 | 240.8 | 252 | 220 | 8065.7 |
| | 3 | 130 | 257.5 | 258 | 253 | 7869.1 |
| | 3 | 160 | 258 | 258 | 258 | 7900.5 |
| 50b | 3 | 10 | 12.1 | 17 | 9 | 2054.2 |
| | 3 | 20 | 24.6 | 27 | 24 | 2100.6 |
| | 3 | 40 | 73.6 | 79 | 69 | 3666.9 |
| | 3 | 50 | 106.3 | 115 | 96 | 6327.4 |
| | 3 | 70 | 171.2 | 181 | 141 | 8383.7 |
| | 3 | 80 | 193.8 | 209 | 160 | 8285.8 |
| | 3 | 100 | 239 | 252 | 228 | 8135.5 |
| | 3 | 130 | 257.6 | 258 | 256 | 7943.5 |
| | 3 | 160 | 258 | 258 | 258 | 7955.5 |
| 50c | 3 | 10 | 4 | 4 | 4 | 2361.1 |
| | 3 | 20 | 11.9 | 16 | 7 | 2381.8 |
| | 3 | 40 | 77 | 84 | 64 | 3814 |
| | 3 | 50 | 104 | 113 | 97 | 6489.9 |
| | 3 | 70 | 177.9 | 204 | 149 | 8638.2 |
| | 3 | 80 | 210.5 | 220 | 199 | 8561 |
| | 3 | 100 | 239.4 | 252 | 233 | 8286.5 |
| | 3 | 130 | 256.8 | 258 | 255 | 8398 |
| | 3 | 160 | 258 | 258 | 258 | 8422 |
| 50d | 3 | 10 | 5 | 5 | 5 | 2132.9 |
| | 3 | 20 | 25 | 25 | 25 | 2345.4 |
| | 3 | 40 | 73.2 | 79 | 68 | 3721.2 |
| | 3 | 50 | 102.6 | 107 | 93 | 6015.9 |
| | 3 | 70 | 163 | 180 | 149 | 8477 |
| | 3 | 80 | 189.8 | 212 | 173 | 8439 |
| | 3 | 100 | 229.5 | 248 | 207 | 8224.6 |
| | 3 | 130 | 256.1 | 258 | 241 | 8098.3 |
| | 3 | 160 | 258 | 258 | 258 | 8061.9 |
| 50e | 3 | 10 | 9.1 | 11 | 6 | 2164.3 |
| | 3 | 20 | 31.9 | 33 | 28 | 2396.7 |
| | 3 | 40 | 80.6 | 88 | 69 | 4538.2 |
| | 3 | 50 | 98.6 | 108 | 80 | 6583 |
| | 3 | 70 | 175.1 | 191 | 149 | 8544.8 |
| | 3 | 80 | 203.1 | 214 | 195 | 8409 |
| | 3 | 100 | 240 | 252 | 216 | 8294.5 |
| | 3 | 130 | 257.6 | 258 | 254 | 8147.7 |
| | 3 | 160 | 258 | 258 | 258 | 8118 |

Table A.1: Results using $\alpha = 1$ and $\beta = 0$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| Data set | $\|K\|$ | $M$ | Average | Max | Min | Average CPU |
|---|---|---|---|---|---|---|
| 50a | 4 | 10 | 16 | 16 | 16 | 2302.5 |
| | 4 | 20 | 24.9 | 31 | 21 | 2182.1 |
| | 4 | 40 | 121.7 | 130 | 106 | 5118.7 |
| | 4 | 50 | 155.4 | 164 | 145 | 7269.8 |
| | 4 | 70 | 207 | 221 | 194 | 10337 |
| | 4 | 80 | 237.4 | 249 | 214 | 10150 |
| | 4 | 100 | 256.6 | 258 | 251 | 9999.4 |
| | 4 | 130 | 258 | 258 | 258 | 9929.9 |
| | 4 | 160 | 258 | 258 | 258 | 9940.4 |
| 50b | 4 | 10 | 13.2 | 17 | 9 | 2170.9 |
| | 4 | 20 | 23.7 | 27 | 19 | 2339.8 |
| | 4 | 40 | 82.8 | 90 | 77 | 4246.6 |
| | 4 | 50 | 124.8 | 133 | 114 | 7683.7 |
| | 4 | 70 | 204.7 | 224 | 191 | 10459 |
| | 4 | 80 | 236.6 | 252 | 222 | 10349 |
| | 4 | 100 | 257.5 | 258 | 255 | 10186 |
| | 4 | 130 | 258 | 258 | 258 | 10084 |
| | 4 | 160 | 258 | 258 | 258 | 10130 |
| 50c | 4 | 10 | 4 | 4 | 4 | 2374.9 |
| | 4 | 20 | 11.7 | 13 | 7 | 2492 |
| | 4 | 40 | 89.3 | 98 | 76 | 4237.4 |
| | 4 | 50 | 136.6 | 147 | 126 | 7782.5 |
| | 4 | 70 | 212.3 | 228 | 183 | 10616 |
| | 4 | 80 | 235.3 | 252 | 228 | 10730 |
| | 4 | 100 | 255.2 | 258 | 252 | 10383 |
| | 4 | 130 | 258 | 258 | 258 | 10377 |
| | 4 | 160 | 258 | 258 | 258 | 10293 |
| 50d | 4 | 10 | 5 | 5 | 5 | 2181.1 |
| | 4 | 20 | 21.5 | 25 | 15 | 2247.9 |
| | 4 | 40 | 84.2 | 88 | 80 | 4228.4 |
| | 4 | 50 | 117.8 | 123 | 110 | 7117.5 |
| | 4 | 70 | 191.9 | 203 | 177 | 10418 |
| | 4 | 80 | 229.8 | 244 | 203 | 10338 |
| | 4 | 100 | 254.9 | 258 | 236 | 10144 |
| | 4 | 130 | 258 | 258 | 258 | 10061 |
| | 4 | 160 | 258 | 258 | 258 | 11618 |
| 50e | 4 | 10 | 9 | 11 | 6 | 2233.3 |
| | 4 | 20 | 31.2 | 33 | 28 | 2484.7 |
| | 4 | 40 | 93.3 | 98 | 87 | 5178.2 |
| | 4 | 50 | 128 | 138 | 113 | 7865.4 |
| | 4 | 70 | 210.9 | 221 | 202 | 10563 |
| | 4 | 80 | 235.2 | 246 | 227 | 10349 |
| | 4 | 100 | 258 | 258 | 258 | 10151 |
| | 4 | 130 | 258 | 258 | 258 | 10116 |
| | 4 | 160 | 258 | 258 | 258 | 10156 |

Table A.2: Results using $\alpha = 1$ and $\beta = 0$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| Data set | $\|K\|$ | $M$ | Average | Max | Min | Average CPU |
|---|---|---|---|---|---|---|
| 50a | 5 | 10 | 16 | 16 | 16 | 2362.8 |
| | 5 | 20 | 27.6 | 31 | 21 | 2324.8 |
| | 5 | 40 | 129.8 | 135 | 124 | 5790.2 |
| | 5 | 50 | 170 | 177 | 158 | 8449 |
| | 5 | 70 | 229 | 239 | 213 | 12068 |
| | 5 | 80 | 252.8 | 257 | 241 | 12102 |
| | 5 | 100 | 258 | 258 | 258 | 11818 |
| | 5 | 130 | 258 | 258 | 258 | 11826 |
| | 5 | 160 | 258 | 258 | 258 | 11851 |
| 50b | 5 | 10 | 14.1 | 17 | 9 | 2294.5 |
| | 5 | 20 | 23.7 | 27 | 19 | 2360.6 |
| | 5 | 40 | 89.9 | 92 | 88 | 4677.8 |
| | 5 | 50 | 146 | 157 | 134 | 8964.3 |
| | 5 | 70 | 238.2 | 249 | 225 | 12473 |
| | 5 | 80 | 255.2 | 258 | 251 | 12292 |
| | 5 | 100 | 258 | 258 | 258 | 12124 |
| | 5 | 130 | 258 | 258 | 258 | 12071 |
| | 5 | 160 | 258 | 258 | 258 | 12077 |
| 50c | 5 | 10 | 4 | 4 | 4 | 2297.1 |
| | 5 | 20 | 11.2 | 16 | 7 | 2168.8 |
| | 5 | 40 | 99.3 | 103 | 89 | 4575.1 |
| | 5 | 50 | 148.5 | 158 | 132 | 8635.3 |
| | 5 | 70 | 233.1 | 247 | 227 | 12307 |
| | 5 | 80 | 249.4 | 255 | 242 | 12225 |
| | 5 | 100 | 257.7 | 258 | 255 | 12146 |
| | 5 | 130 | 258 | 258 | 258 | 12144 |
| | 5 | 160 | 258 | 258 | 258 | 12037 |
| 50d | 5 | 10 | 5 | 5 | 5 | 2247.3 |
| | 5 | 20 | 21.6 | 25 | 20 | 2287.3 |
| | 5 | 40 | 91.3 | 93 | 90 | 4824.4 |
| | 5 | 50 | 133.1 | 140 | 113 | 8480.9 |
| | 5 | 70 | 226.4 | 235 | 215 | 12457 |
| | 5 | 80 | 245.6 | 255 | 234 | 12392 |
| | 5 | 100 | 258 | 258 | 258 | 12240 |
| | 5 | 130 | 258 | 258 | 258 | 12178 |
| | 5 | 160 | 258 | 258 | 258 | 12189 |
| 50e | 5 | 10 | 8 | 11 | 6 | 2241 |
| | 5 | 20 | 29.9 | 33 | 26 | 2532.9 |
| | 5 | 40 | 100.7 | 102 | 97 | 5875.8 |
| | 5 | 50 | 142.4 | 154 | 133 | 9308.6 |
| | 5 | 70 | 230.2 | 238 | 222 | 12339 |
| | 5 | 80 | 252.6 | 258 | 234 | 12260 |
| | 5 | 100 | 258 | 258 | 258 | 12146 |
| | 5 | 130 | 258 | 258 | 258 | 12139 |
| | 5 | 160 | 258 | 258 | 258 | 12070 |

Table A.3: Results using $\alpha = 1$ and $\beta = 0$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| **Data set** | $\|K\|$ | $M$ | **Average** | **Max** | **Min** | **Average CPU** |
|---|---|---|---|---|---|---|
| 100a | 3 | 10 | 11 | 11 | 11 | 2398.9 |
| | 3 | 20 | 35.5 | 38 | 34 | 3003.3 |
| | 3 | 40 | 89.6 | 102 | 66 | 9508.3 |
| | 3 | 50 | 116.8 | 126 | 90 | 19074 |
| | 3 | 70 | 152.6 | 174 | 133 | 28235 |
| | 3 | 80 | 181.6 | 195 | 165 | 27764 |
| | 3 | 100 | 230.8 | 266 | 210 | 26520 |
| | 3 | 130 | 266.4 | 304 | 218 | 25725 |
| | 3 | 160 | 335.5 | 379 | 301 | 25165 |
| 100b | 3 | 10 | 17.5 | 23 | 13 | 3381.5 |
| | 3 | 20 | 43.1 | 47 | 40 | 4202.5 |
| | 3 | 40 | 97.7 | 106 | 86 | 7283 |
| | 3 | 50 | 113.1 | 130 | 98 | 12593 |
| | 3 | 70 | 156.1 | 171 | 140 | 17413 |
| | 3 | 80 | 183.9 | 197 | 168 | 18224 |
| | 3 | 100 | 221.9 | 245 | 200 | 16983 |
| | 3 | 130 | 278.5 | 325 | 235 | 16522 |
| | 3 | 160 | 327.2 | 376 | 284 | 15356 |
| 100c | 3 | 10 | 28 | 30 | 20 | 4146.9 |
| | 3 | 20 | 53.3 | 54 | 47 | 4493.6 |
| | 3 | 40 | 98.8 | 104 | 93 | 7775.1 |
| | 3 | 50 | 128.5 | 143 | 115 | 12966 |
| | 3 | 70 | 178 | 201 | 143 | 17580 |
| | 3 | 80 | 205.4 | 233 | 173 | 16659 |
| | 3 | 100 | 242.5 | 282 | 210 | 16618 |
| | 3 | 130 | 295.5 | 340 | 248 | 15192 |
| | 3 | 160 | 352.9 | 392 | 316 | 15150 |
| 100d | 3 | 10 | 20.3 | 22 | 12 | 3925.1 |
| | 3 | 20 | 43.5 | 45 | 40 | 4427.8 |
| | 3 | 40 | 83 | 91 | 68 | 6736.5 |
| | 3 | 50 | 96.7 | 111 | 83 | 12487 |
| | 3 | 70 | 145.6 | 163 | 115 | 17324 |
| | 3 | 80 | 172.6 | 182 | 142 | 17978 |
| | 3 | 100 | 215.9 | 232 | 198 | 16567 |
| | 3 | 130 | 265.8 | 306 | 228 | 15795 |
| | 3 | 160 | 314.3 | 335 | 255 | 15609 |
| 100e | 3 | 10 | 8 | 8 | 8 | 4633.4 |
| | 3 | 20 | 37.7 | 38 | 35 | 4814.9 |
| | 3 | 40 | 96.5 | 109 | 86 | 9371.8 |
| | 3 | 50 | 123.3 | 141 | 105 | 12709 |
| | 3 | 70 | 160.7 | 185 | 107 | 18782 |
| | 3 | 80 | 195.1 | 215 | 175 | 18814 |
| | 3 | 100 | 232.4 | 258 | 207 | 17216 |
| | 3 | 130 | 288.6 | 315 | 242 | 16074 |
| | 3 | 160 | 354.8 | 377 | 336 | 16341 |

Table A.4: Results using $\alpha = 1$ and $\beta = 0$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| Data set | $\|K\|$ | $M$ | Average | Max | Min | Average CPU |
|---|---|---|---|---|---|---|
| 100a | 4 | 10 | 11 | 11 | 11 | 2404.5 |
| | 4 | 20 | 42.7 | 44 | 38 | 3196.8 |
| | 4 | 40 | 115.2 | 126 | 102 | 11631 |
| | 4 | 50 | 155 | 170 | 140 | 23627 |
| | 4 | 70 | 206.9 | 227 | 181 | 35611 |
| | 4 | 80 | 224.8 | 246 | 191 | 34632 |
| | 4 | 100 | 277.3 | 316 | 235 | 33560 |
| | 4 | 130 | 362.6 | 380 | 314 | 31908 |
| | 4 | 160 | 411.7 | 452 | 370 | 31406 |
| 100b | 4 | 10 | 16.8 | 23 | 13 | 3518.7 |
| | 4 | 20 | 50.6 | 54 | 45 | 4461.5 |
| | 4 | 40 | 118 | 131 | 107 | 8125.3 |
| | 4 | 50 | 142 | 159 | 125 | 14834 |
| | 4 | 70 | 206.6 | 223 | 168 | 20674 |
| | 4 | 80 | 209.8 | 243 | 182 | 20979 |
| | 4 | 100 | 273.3 | 300 | 234 | 20222 |
| | 4 | 130 | 345.9 | 373 | 301 | 18515 |
| | 4 | 160 | 404.1 | 433 | 383 | 17062 |
| 100c | 4 | 10 | 25.8 | 30 | 20 | 3683.4 |
| | 4 | 20 | 61.5 | 63 | 54 | 4353.6 |
| | 4 | 40 | 118.1 | 134 | 111 | 8298.9 |
| | 4 | 50 | 154.2 | 174 | 132 | 14958 |
| | 4 | 70 | 232.5 | 256 | 213 | 21281 |
| | 4 | 80 | 261.1 | 284 | 221 | 19116 |
| | 4 | 100 | 296.1 | 334 | 246 | 18789 |
| | 4 | 130 | 378.8 | 435 | 333 | 17720 |
| | 4 | 160 | 436.8 | 455 | 402 | 16923 |
| 100d | 4 | 10 | 16.9 | 21 | 12 | 3631.5 |
| | 4 | 20 | 51.6 | 52 | 51 | 4367.4 |
| | 4 | 40 | 99.5 | 115 | 89 | 7517 |
| | 4 | 50 | 129.2 | 136 | 111 | 14742 |
| | 4 | 70 | 186.7 | 203 | 162 | 20872 |
| | 4 | 80 | 215.1 | 249 | 188 | 21440 |
| | 4 | 100 | 281.5 | 307 | 249 | 19357 |
| | 4 | 130 | 352.3 | 378 | 325 | 18277 |
| | 4 | 160 | 413.9 | 448 | 372 | 17440 |
| 100e | 4 | 10 | 8 | 8 | 8 | 4694.2 |
| | 4 | 20 | 44.4 | 46 | 41 | 5669 |
| | 4 | 40 | 120.4 | 130 | 112 | 9416.6 |
| | 4 | 50 | 154.8 | 176 | 107 | 15486 |
| | 4 | 70 | 218.9 | 245 | 198 | 21922 |
| | 4 | 80 | 250.8 | 271 | 230 | 22476 |
| | 4 | 100 | 282.8 | 307 | 254 | 19784 |
| | 4 | 130 | 366.1 | 402 | 342 | 19978 |
| | 4 | 160 | 415.7 | 439 | 395 | 17509 |

Table A.5: Results using $\alpha = 1$ and $\beta = 0$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| **Data set** | $\|K\|$ | $M$ | **Average** | **Max** | **Min** | **Average CPU** |
|---|---|---|---|---|---|---|
| 100a | 5 | 10 | 11 | 11 | 11 | 4114.6 |
| | 5 | 20 | 45.2 | 46 | 44 | 4704.7 |
| | 5 | 40 | 139.9 | 151 | 125 | 10074 |
| | 5 | 50 | 180.2 | 204 | 162 | 17528 |
| | 5 | 70 | 237.7 | 259 | 210 | 26868 |
| | 5 | 80 | 273.6 | 301 | 255 | 24430 |
| | 5 | 100 | 333 | 366 | 299 | 22919 |
| | 5 | 130 | 429.4 | 446 | 406 | 20914 |
| | 5 | 160 | 470.7 | 502 | 416 | 20513 |
| 100b | 5 | 10 | 17.3 | 23 | 13 | 3973.5 |
| | 5 | 20 | 56.5 | 59 | 52 | 4989.3 |
| | 5 | 40 | 141.7 | 152 | 119 | 9463.7 |
| | 5 | 50 | 164.8 | 185 | 143 | 17603 |
| | 5 | 70 | 249.5 | 276 | 226 | 24986 |
| | 5 | 80 | 276.6 | 301 | 255 | 25792 |
| | 5 | 100 | 335.7 | 358 | 298 | 22461 |
| | 5 | 130 | 408.3 | 431 | 378 | 21481 |
| | 5 | 160 | 465.7 | 494 | 433 | 20413 |
| 100c | 5 | 10 | 24.6 | 30 | 20 | 3588.4 |
| | 5 | 20 | 65.4 | 71 | 63 | 4402.7 |
| | 5 | 40 | 131.6 | 152 | 109 | 8891.5 |
| | 5 | 50 | 183.2 | 197 | 173 | 17426 |
| | 5 | 70 | 265.9 | 283 | 242 | 25305 |
| | 5 | 80 | 302.2 | 327 | 274 | 23638 |
| | 5 | 100 | 362 | 404 | 321 | 22466 |
| | 5 | 130 | 433.9 | 459 | 420 | 21035 |
| | 5 | 160 | 488.3 | 504 | 459 | 19577 |
| 100d | 5 | 10 | 17 | 21 | 12 | 3532.4 |
| | 5 | 20 | 53.7 | 55 | 52 | 4388.1 |
| | 5 | 40 | 113.8 | 125 | 108 | 8386.2 |
| | 5 | 50 | 150.1 | 169 | 134 | 18106 |
| | 5 | 70 | 231.8 | 246 | 208 | 27403 |
| | 5 | 80 | 265.3 | 288 | 239 | 24810 |
| | 5 | 100 | 331.3 | 355 | 296 | 23208 |
| | 5 | 130 | 422.7 | 451 | 395 | 20982 |
| | 5 | 160 | 479.9 | 502 | 455 | 19146 |
| 100e | 5 | 10 | 8 | 8 | 8 | 4814 |
| | 5 | 20 | 53 | 56 | 49 | 5314 |
| | 5 | 40 | 140.3 | 148 | 127 | 10134 |
| | 5 | 50 | 182.2 | 200 | 137 | 17948 |
| | 5 | 70 | 245 | 273 | 211 | 27232 |
| | 5 | 80 | 294.7 | 309 | 285 | 25393 |
| | 5 | 100 | 342.8 | 372 | 321 | 24342 |
| | 5 | 130 | 417.9 | 432 | 357 | 22018 |
| | 5 | 160 | 476 | 502 | 457 | 22587 |

Table A.6: Results using $\alpha = 1$ and $\beta = 0$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| **Data set** | $\|K\|$ | $M$ | **Average** | **Max** | **Min** | **Average CPU** |
|---|---|---|---|---|---|---|
| 3_50_$a$ | 3 | 5 | 59 | 60 | 50 | 4482.7 |
| | 3 | 10 | 102.8 | 111 | 90 | 6377.9 |
| | 3 | 15 | 168 | 171 | 141 | 8565 |
| | 3 | 20 | 231.8 | 232 | 231 | 10623 |
| | 3 | 25 | 286.9 | 301 | 222 | 11628 |
| | 3 | 30 | 361 | 361 | 361 | 11268 |
| | 3 | 35 | 394.4 | 411 | 371 | 10987 |
| | 3 | 40 | 424.8 | 432 | 371 | 10325 |
| | 3 | 50 | 430 | 432 | 412 | 10065 |
| 3_50_$b$ | 3 | 5 | 55 | 55 | 55 | 4833.6 |
| | 3 | 10 | 117 | 120 | 100 | 6275.6 |
| | 3 | 15 | 171 | 180 | 140 | 8560.7 |
| | 3 | 20 | 249 | 250 | 240 | 10993 |
| | 3 | 25 | 292.9 | 301 | 230 | 12022 |
| | 3 | 30 | 356.9 | 361 | 330 | 11570 |
| | 3 | 35 | 401.8 | 411 | 370 | 11125 |
| | 3 | 40 | 428.9 | 432 | 401 | 10317 |
| | 3 | 50 | 423 | 432 | 372 | 10403 |
| 3_50_$c$ | 3 | 5 | 55 | 55 | 55 | 4843.8 |
| | 3 | 10 | 104.5 | 111 | 90 | 6605.6 |
| | 3 | 15 | 179 | 181 | 171 | 8756.9 |
| | 3 | 20 | 228.1 | 231 | 222 | 11517 |
| | 3 | 25 | 299.2 | 300 | 292 | 12029 |
| | 3 | 30 | 359 | 360 | 350 | 11760 |
| | 3 | 35 | 402.9 | 411 | 370 | 11730 |
| | 3 | 40 | 428 | 432 | 392 | 11412 |
| | 3 | 50 | 429 | 432 | 402 | 10319 |

Table A.7: Results using $\alpha = 1$ and $\beta = 15$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| **Data set** | $\|K\|$ | $M$ | **Average** | **Max** | **Min** | **Average CPU** |
|---|---|---|---|---|---|---|
| 4_50_$a$ | 4 | 4 | 72.5 | 75 | 65 | 6816.4 |
| | 4 | 8 | 129.6 | 131 | 120 | 9613.7 |
| | 4 | 12 | 190.8 | 201 | 171 | 17074 |
| | 4 | 16 | 272 | 281 | 211 | 14932 |
| | 4 | 20 | 359.9 | 361 | 351 | 14440 |
| | 4 | 24 | 423 | 432 | 352 | 13535 |
| | 4 | 28 | 431.9 | 432 | 431 | 12780 |
| | 4 | 30 | 430.9 | 432 | 421 | 12427 |
| | 4 | 35 | 426 | 432 | 392 | 12669 |
| 4_50_$b$ | 4 | 4 | 71.5 | 75 | 60 | 7025.3 |
| | 4 | 8 | 144 | 150 | 120 | 9696.4 |
| | 4 | 12 | 206.9 | 211 | 191 | 16331 |
| | 4 | 16 | 273.4 | 290 | 241 | 14673 |
| | 4 | 20 | 364.1 | 370 | 331 | 14050 |
| | 4 | 24 | 423.9 | 432 | 372 | 12864 |
| | 4 | 28 | 423.7 | 432 | 400 | 13007 |
| | 4 | 30 | 417.9 | 432 | 341 | 12435 |
| | 4 | 35 | 430.9 | 432 | 421 | 12172 |
| 4_50_$c$ | 4 | 4 | 75 | 80 | 60 | 6975.3 |
| | 4 | 8 | 156 | 160 | 140 | 9060.5 |
| | 4 | 12 | 229.9 | 231 | 220 | 15812 |
| | 4 | 16 | 286.4 | 292 | 261 | 14074 |
| | 4 | 20 | 361.5 | 372 | 320 | 13112 |
| | 4 | 24 | 423.9 | 432 | 401 | 12542 |
| | 4 | 28 | 428 | 432 | 412 | 12264 |
| | 4 | 30 | 428.9 | 432 | 412 | 12323 |
| | 4 | 35 | 432 | 432 | 432 | 11720 |

Table A.8: Results using $\alpha = 1$ and $\beta = 15$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| **Data set** | $\|K\|$ | $M$ | **Average** | **Max** | **Min** | **Average CPU** |
|---|---|---|---|---|---|---|
| 4_100_$a$ | 4 | 10 | 183 | 190 | 170 | 11610 |
| | 4 | 20 | 362 | 380 | 300 | 44446 |
| | 4 | 25 | 462.6 | 471 | 430 | 69766 |
| | 4 | 30 | 544.8 | 561 | 431 | 57475 |
| | 4 | 35 | 628 | 661 | 600 | 53055 |
| | 4 | 40 | 697.2 | 750 | 563 | 47969 |
| | 4 | 45 | 742.3 | 814 | 582 | 44791 |
| | 4 | 50 | 835.2 | 904 | 713 | 44777 |
| | 4 | 60 | 875.3 | 905 | 814 | 41065 |
| 4_100_$b$ | 4 | 10 | 182 | 190 | 150 | 11757 |
| | 4 | 20 | 377 | 380 | 360 | 39682 |
| | 4 | 25 | 440 | 470 | 370 | 65734 |
| | 4 | 30 | 566.6 | 571 | 551 | 58835 |
| | 4 | 35 | 651.2 | 670 | 621 | 56639 |
| | 4 | 40 | 684.5 | 751 | 540 | 47337 |
| | 4 | 45 | 790.2 | 841 | 750 | 44403 |
| | 4 | 50 | 842.5 | 904 | 795 | 42063 |
| | 4 | 60 | 882.2 | 905 | 834 | 41686 |
| 4_100_$c$ | 4 | 10 | 190 | 190 | 190 | 11702 |
| | 4 | 20 | 378 | 390 | 310 | 42901 |
| | 4 | 25 | 463 | 470 | 450 | 69121 |
| | 4 | 30 | 548 | 580 | 450 | 60838 |
| | 4 | 35 | 644.3 | 661 | 580 | 53120 |
| | 4 | 40 | 681.4 | 761 | 590 | 47479 |
| | 4 | 45 | 793.6 | 841 | 721 | 46224 |
| | 4 | 50 | 820.8 | 904 | 721 | 43629 |
| | 4 | 60 | 883.2 | 905 | 853 | 41369 |

Table A.9: Results using $\alpha = 1$ and $\beta = 15$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

| **Data set** | $\|K\|$ | $M$ | **Average** | **Max** | **Min** | **Average CPU** |
|---|---|---|---|---|---|---|
| 3_100_$a$ | 3 | 10 | 118 | 121 | 111 | 6121.4 |
| | 3 | 20 | 231.1 | 240 | 161 | 12729 |
| | 3 | 30 | 358.1 | 370 | 281 | 23002 |
| | 3 | 35 | 414.7 | 431 | 331 | 30998 |
| | 3 | 40 | 492.1 | 500 | 480 | 34771 |
| | 3 | 45 | 521.3 | 560 | 410 | 41077 |
| | 3 | 50 | 587.4 | 620 | 520 | 43730 |
| | 3 | 60 | 691.7 | 732 | 631 | 41873 |
| | 3 | 75 | 805.6 | 863 | 713 | 38226 |
| 3_100_$b$ | 3 | 10 | 130 | 130 | 130 | 6071.3 |
| | 3 | 20 | 246 | 250 | 220 | 12023 |
| | 3 | 30 | 359 | 380 | 290 | 23078 |
| | 3 | 35 | 410 | 430 | 340 | 28914 |
| | 3 | 40 | 475 | 500 | 390 | 36920 |
| | 3 | 45 | 540.2 | 560 | 411 | 37747 |
| | 3 | 50 | 594 | 620 | 550 | 44592 |
| | 3 | 60 | 681.8 | 741 | 622 | 40227 |
| | 3 | 75 | 784.3 | 874 | 690 | 34663 |
| 3_100_$c$ | 3 | 10 | 112 | 120 | 100 | 5802.6 |
| | 3 | 20 | 237 | 240 | 230 | 11083 |
| | 3 | 30 | 328 | 350 | 270 | 22702 |
| | 3 | 35 | 411 | 420 | 380 | 27548 |
| | 3 | 40 | 456 | 480 | 340 | 35322 |
| | 3 | 45 | 526.3 | 551 | 400 | 39322 |
| | 3 | 50 | 563.5 | 601 | 440 | 38953 |
| | 3 | 60 | 685.4 | 732 | 622 | 39509 |
| | 3 | 75 | 811.2 | 881 | 753 | 34580 |

Table A.10: Results using $\alpha = 1$ and $\beta = 15$. Average objective, best calculated objective, worst calculated objective and average run times in mille seconds are displayed.

# Appendix B

# Solution Types

A few types of solutions are presented here and then analyzed with SWOT analysis. There internal benefactors of the project are: the student, ALCAN and the professors. The external benefactors are:The employees of ALCAN, Hópbílar (or other transport companies) and the general public.
**Type 1**: Use current pickup points along with new ones (predefined, such as local bus stops). Estimate the importance of each pickup point by the number of people living close to it, the amount of parking and connection to local transit system. Buses from Hópbílar are used to pick up employees.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general solution that takes into account employee turnover. Works all year round, 24 hours a day. This solution is not too simple to be considered a exam project. | Not ALCAN's desired solution. In this solution new nodes without a predefined location cannot be used. Hard to estimate the general population of an area. |
| **External** | Decreases travel time. | Decreases profit for Hópbílar. Decreases the current amount of service provided by ALCAN. |

Transportation in this solution is provided by Hópbílar.

**Type 2**: Same as type 1 except importance of pickup points is decided by the number of employees that live close to them.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. Works all year round, 24 hours a day. Not to simple to be considered an exam project. Relatively simple to program and there fore a good candidate for the first solution. | This is a special solution that does not take into account employee turnover. In this solution new nodes without a predefined location cannot be used. Not ALCAN's desired solution. |
| **External** | Decreases travel time. | Decreases profit for Hópbílar. Decreases the current amount of service provided by ALCAN. |

Transportation in this solution is provided by Hópbílar.

**Type 3** Same as type 2 except a soft wear, such as ShorTrec from AGR hf., is used to determine the bus routes. A new route can be calculated as often as ALCAN desires.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. Works all year round, 24 hours a day. A general solution that takes into account employee turnover. | This solution depends on a 3 party program. This solution is too simple to be considered an exam project unless the 3 party soft wear is programmed by the student. Not ALCAN's desired solution. |
| **External** | Decreases travel time. Increases the profit for the provider of the new soft wear. | Decreases profit for Hópbílar. Decreases the current amount of service provided by ALCAN. |

Transportation in this solution is provided by Hópbílar.

**Type 4**: Uses the local transit system, buses, to pickup employees and return them.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. This solution gives good publicity for ALCAN by increasing the use of the local bus system. A general solution that takes into account employee turnover. Solves overtime problem. | This solution is too simple to be considered an exam project. Doesn't work all year round, 24 hours a day. Not ALCAN's desired solution. |
| **External** | Decreases travel time. No employee will have to walk further than 600m. Possible for employees to use outside of working hours. Increases use of the local bus system. | Removes Hópbílar from the picture. Decreases the current amount of service provided by ALCAN. Strætó will have to put up a new bus stop in Straumsvík. |

Transportation in this solution is provided by Strætó.

**Type 5** : Car pooling. Each car will be given a driving diary and receive a payment for gas used at the end of the month. It would be necessary to right a program that would put five optimal people together as a part of a car pooling team.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. Might solve overtime problem. Not to simple to be considered an exam project. This solution works all year round, 24 hours a day. | A special solution that does not take into account employee turnover. Might be misused by employees, who could log more kilometers than they actually drove. Not ALCAN's desired solution. |
| **External** | Decreases travel time. No employee will have to walk, they are picked up at there doorstep. | Removes Hópbílar from the picture. Decreases the current amount of service provided by ALCAN. Employees depend on one another to be at work on time. Not every one owns a car. |

Transportation in this solution is provided by Employees.

**Type 6** : Driving grant. Each employee would receive an increase in pay to compensate for the lack of buses. The employees would then drive themselves to work.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A special solution that takes into account employee turnover. Solves overtime problem. This solution works all year round, 24 hours a day. | Too simple to be considered an exam project. Not ALCAN's desired solution. |
| **External** | Decreases travel time. Increases employee pay (which is always popular). | Removes Hópbílar from the picture. Decreases the current amount of service provided by ALCAN. |

Transportation in this solution is provided by Employees.

**Type 7** : Car pooling with taxis. A taxi would pickup employees and return them. Each taxi would be filled with passengers. A program would tell the taxi service where and when to pick up an employee.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Possibly solves overtime problem. This solution works all year round, 24 hours a day. Not too simple too be considered an exam project. | A special solution that does not take into account employee turnover. Cost of this solution is unknown. Not ALCAN's desired solution. |
| **External** | Decreases travel time. Service is increased as all employees are picked up on there doorstep. Profit for taxi service is increased. | Removes Hópbílar from the picture. |

Transportation in this solution is provided by a taxi service.

**Type 8** : Same as type 1 except the pickup points would be calculated so that there location was optimal and not from predetermined points.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general solution that takes into account employee turnover. This solution works all year round, 24 hours a day. Not too simple too be considered an exam project. ALCAN's desired solution. | Hard to estimate general population of an area. Of all the solutions likely to be the most complicated to formulate. |
| **External** | Decreases travel time. | Service is decreased. Hópbílar's profit is decreased. |

Transportation in this solution is provided by a Hópbílar.

### B.0.1 Combined solutions

**Combo 1** : Type 1 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 1 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general solution that takes into account employee turnover. This solution works all year round, 24 hours a day. Not too simple too be considered an exam project. Solves the overtime problem during day/evening on non holidays. Good publicity for ALCAN as the public transport system gains more users. | Not ALCAN's desired solution. In this solution new nodes without predefined locations can not be defined. Hard to estimate general population of an area. |
| **External** | No employee will have to walk further than 600m during day/evening on non holidays. Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. | Service is decreased. Hópbílar's profit is decreased. Strætó will have to build a new bus stop in Straumsvík. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 2** : Type 2 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 2 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general and special solution that takes into account, during day/evening on non holidays, employee turnover. This solution works all year round, 24 hours a day. Not too simple too be considered an exam project. Partly solves the overtime problem. Good publicity for ALCAN as the public transport system gains more users. | A general and special solution that does not take into account, during night or on holidays, employee turnover. Not ALCAN's desired solution. In this solution new nodes without predefined locations can not be defined. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m during day/evening on non holidays. | Service is decreased. Hópbílar's profit is decreased. Strætó will have to build a new bus stop in Straumsvík. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 3** : Type 3 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 3 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general and special solution that takes into account employee turnover. This solution works all year round, 24 hours a day. Partly solves the overtime problem. Good publicity for ALCAN as the public transport system gains more users. | Not ALCAN's desired solution. This solution is too simple to be considered an exam project unless the 3 party soft wear is programmed by the author of the project. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m during day/evening on non holidays. | Service is decreased. Hópbílar's profit is decreased. Strætó will have to build a new bus stop in Straumsvík. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 4** : Type 5 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 5 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general and special solution that takes into account, during day/evening on non holidays, employee turnover. This solution works all year round, 24 hours a day. Partly (even completely) solves the overtime problem. Good publicity for ALCAN as the public transport system gains more users. | A general and special solution that does not take into account, during night or on holidays, employee turnover. Not ALCAN's desired solution. Might be misused by employees, who could log more kilometers than they actually have driven. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m during day/evening on non holidays. | Service is decreased. Removes Hópbílar from the picture. Strætó will have to build a new bus stop in Straumsvík. Employees depend on one another to be at work on time. Not every one owns a car. |

Transportation in this solution is provided by a Employees and Strætó.

**Combo 5** : Type 6 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 6 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general solution that takes into account employee turnover. This solution works all year round, 24 hours a day. Solves the overtime problem. Good publicity for ALCAN as the public transport system gains more users. | Not ALCAN's desired solution. This solution is too simple to be considered an exam project. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. Employees receive an increase in pay. No employee will have to walk further than 600m during day/evening on non holidays. | Service is decreased. Removes Hópbílar from the picture. Strætó will have to build a new bus stop in Straumsvík. Not every one owns a car. |

Transportation in this solution is provided by Employees and Strætó.

**Combo 6** : Type 7 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 7 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. A general solution,during day/evening on non holidays, that takes into account employee turnover. This solution works all year round, 24 hours a day. Solves the overtime problem. Good publicity for ALCAN as the public transport system gains more users. Not too simple to be considered an exam project. | A special solution, at night and on holidays, that does not take into account employee turnover. Not ALCAN's desired solution. Cost is an unknown factor. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. Service is increased during night and holidays. No employee will have to walk further than 600m during day/evening on non holidays. Increased revenue for the taxi service. | Service is decreased during day/evening on non holidays. Removes Hópbílar from the picture. Strætó will have to build a new bus stop in Straumsvík. |

Transportation in this solution is provided by a taxi service and Strætó.

**Combo 7** : Type 8 and type 4.
**Description** : Use solution type 4 when it is possible, during daytime on non holidays, and solution type 8 when type 4 is not available.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. A general solution that takes into account employee turnover. This solution works all year round, 24 hours a day. Partly solves the overtime problem. Good publicity for ALCAN as the public transport system gains more users. Not too simple to be considered an exam project. | Hard to estimate general population of an area. Not ALCAN's desired solution. Likely a complicated solution. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m during day/evening on non holidays . | Service is decreased. Hópbílar's profit is decreased. Strætó will have to build a new bus stop in Straumsvík. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 8** : Type 5 and type 6.
**Description** : Type 5 but instead of using the driving diaries, employees would receive an increase in pay, type 6, for driving there fellow coworkers to work.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases cost. Decreases travel time. This solution works all year round, 24 hours a day. Could solve the overtime problem. Not too simple to be considered an exam project. | Not ALCAN's desired solution. A special solution that does not take into account employee turnover. |
| **External** | Decreases travel time. Employees are picked up at there doorstep. Employees receive pay increase. | Service provided by ALCAN is decreased. Hópbílar are removed from the picture. Employees depend on one another to be at work on time. Not every one owns a car. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 9** : Type 7 and type 6.
**Description** : Solution type 7 would be used but instead of ALCAN paying the taxi service it would increase workers pay. Employees would then use that pay increase to pay for the taxies.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. This solution works all year round, 24 hours a day. Could solve the overtime problem. Not too simple to be considered an exam project. | Not ALCAN's desired solution. Cost is unknown, likely high. A special solution that does not take into account employee turnover. |
| **External** | Decreases travel time. Employees are picked up at there doorstep. Employees receive pay increase. Service is increased. Increased revenue for the taxi service. | Hópbílar are removed from the picture. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 10** : Extreme solution using type 1 and type 4.
**Description** : Solve solution type 1 with as few routes and pickup points as possible. Employees then use local buses to get to those points, type 4.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. A general solution that takes into account employee turnover. Good publicity for ALCAN as the public transport system gains more users. Not too simple to be considered an exam project. | Not ALCAN's desired solution. Does not work all year round, 24 hours a day. Cost might not be decreased. In this solution new nodes without predefined locations can not be defined. Hard to estimate general population of an area. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m. | Service is decreased. Hópbílar's profit is decreased. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 11** : Extreme solution using type 2 and type 4.
**Description** : Solve solution type 2 with as few routes and pickup points as possible. Employees then use local buses to get to those points, type 4.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. A general and special solution that takes into account employee turnover. Good publicity for ALCAN as the public transport system gains more users. Not too simple to be considered an exam project. | Not ALCAN's desired solution. Does not work all year round, 24 hours a day. Cost might not be decreased. In this solution new nodes without predefined locations can not be defined. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m. | Service is decreased. Hópbílar's profit is decreased. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 12** : Extreme solution using type 3 and type 4.
**Description** : Solve solution type 3 with as few routes and pickup points as possible. Employees then use local buses to get to those points, type 4.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. A general and special solution that takes into account employee turnover. Good publicity for ALCAN as the public transport system gains more users. | Depend on a third party program. Too simple to be considered an exam project, unless the third party program is program by the author of the project. Not ALCAN's desired solution. Does not work all year round, 24 hours a day. Cost might not be decreased. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m. Increased revenue for the taxi service. | Service is decreased. Hópbílar's profit is decreased. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 13** : Extreme solution using type 8 and type 4.
**Description** : Solve solution type 8 with as few routes and pickup points as possible. Employees then use local buses to get to those points, type 4.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. A general solution that takes into account employee turnover. Good publicity for ALCAN as the public transport system gains more users. Not too simple to be considered an exam project. | Not ALCAN's desired solution. Does not work all year round, 24 hours a day. Cost might not be decreased. Likely a complicated solution. |
| **External** | Decreases travel time. Employees can use the public buses when they are not at work. More users for the public transport system. No employee will have to walk further than 600m. | Service is decreased. Hópbílar's profit is decreased. |

Transportation in this solution is provided by a Hópbílar and Strætó.

**Combo 14** : Extreme solution using type 1 and type 5.
**Description** : Solve solution type 1 with as few routes and pickup points as possible. Employees then use car pooling to get to those points, type 5.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. Not too simple to be considered an exam project. | A general and special solution that does not takes into account employee turnover. Not ALCAN's desired solution. In this solution new nodes without predefined locations can not be defined. Might be misused by employees who could log more kilometers then they actually have driven. Hard to estimate general population of an area. Cost might not be decreased. |
| **External** | Decreases travel time. Employee picked up at doorstep. | Service is decreased. Hópbílar's profit is decreased. Employees depend on one another to catch the bus. Not every one owns a car. |

Transportation in this solution is provided by Hópbílar and employees.

**Combo 15** : Extreme solution using type 2 and type 5.
**Description** : Solve solution type 2 with as few routes and pickup points as possible. Employees then use car pooling to get to those points, type 5.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. Not too simple to be considered an exam project. | A special that does not takes into account employee turnover. Not ALCAN's desired solution. Might be misused by employees who could log more kilometers then they actually have driven. In this solution new nodes without predefined locations can not be defined. Cost might not be decreased. |
| **External** | Decreases travel time. Employee picked up at doorstep. | Service is decreased. Hópbílar's profit is decreased. Employees depend on one another to catch the bus. Not every one owns a car. |

Transportation in this solution is provided by Hópbílar and employees.

**Combo 16** : Extreme solution using type 3 and type 5.
**Description** : Solve solution type 3 with as few routes and pickup points as possible. Employees then use car pooling to get to those points, type 5.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. A special solution that takes into account employee turnover. This solution is not too simple to be considered an exam project. | Not ALCAN's desired solution. Depends on a third party program. Might be misused by employees who could log more kilometers then they actually have driven. Cost might not be decreased. |
| **External** | Decreases travel time. Employee picked up at doorstep. Increases profit for the soft wear provider. | Service is decreased. Hópbílar's profit is decreased. Employees depend on one another to catch the bus. Not every one owns a car. |

Transportation in this solution is provided by employees and Hópbílar.

**Combo 17** : Extreme solution using type 8 and type 5.
**Description** : Solve solution type 8 with as few routes and pickup points as possible. Employees then use car pooling to get to those points, type 5.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. This solution is not too simple to be considered an exam project. | Not ALCAN's desired solution. A general and special solution that does not take into account employee turnover. Might be misused by employees who could log more kilometers then they actually have driven. Cost might not be decreased. |
| **External** | Decreases travel time. Employee picked up at doorstep. | Service is decreased. Hópbílar's profit is decreased. Employees depend on one another to catch the bus. Not every one owns a car. Likely a complicated solution. |

Transportation in this solution is provided by employees and Hópbílar.

**Combo 18** : Extreme solution using type 1 and type 6.
**Description** : Solve solution type 1 with as few routes and pickup points as possible. Employees then receive an increase in monthly pay to be used to get to said points, type 6.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. A general solution that takes into account employee turnover. This solution is not too simple to be considered an exam project. | Not ALCAN's desired solution. New nodes without predefined locations cannot be used. Hard to estimate a general population of an area. Cost might not be decreased. |
| **External** | Decreases travel time. Increases pay for employees. | Service is decreased. Hópbílar's profit is decreased. Not every one owns a car. |

Transportation in this solution is provided by employees and Hópbílar.

**Combo 19** : Extreme solution using type 2 and type 6.
**Description** : Solve solution type 2 with as few routes and pickup points as possible. Employees then receive an increase in monthly pay to be used to get to said points, type 6.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. A special solution that partly takes into account employee turnover. This solution is not too simple to be considered an exam project. | Not ALCAN's desired solution. New nodes without predefined locations cannot be used. Cost might not be decreased. |
| **External** | Decreases travel time. Increases pay for employees. | Service is decreased. Hópbílar's profit is decreased. Not every one owns a car. |

Transportation in this solution is provided by employees and Hópbílar.

**Combo 20** : Extrema solution using type 3 and type 6.
**Description** : Solve solution type 3 with as few routes and pickup points as possible. Employees then receive an increase in monthly pay to be used to get to said points, type 6.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. A special solution that takes into account employee turnover. | Too simple to be considered an exam project, unless the third party program is program by the author of the project. Not ALCAN's desired solution. Depends on a third party program. Cost might not be decreased. |
| **External** | Decreases travel time. Increases pay. Increases profit for software provider. | Service is decreased. Hópbílar's profit is decreased. Not every one owns a car. |

Transportation in this solution is provided by employees and Hópbílar.

**Combo 21** : Extreme solution using type 8 and type 6.
**Description** : Solve solution type 8 with as few routes and pickup points as possible. Employees then receive an increase in monthly pay to be used to get to said points, type 6.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. A general solution that takes into account employee turnover. This solution is not too simple to be considered an exam project. | Not ALCAN's desired solution. Hard to estimate a general population of an area. Cost might not be decreased. Likely a complicated solution. |
| **External** | Decreases travel time. Increases pay for employees. | Service is decreased. Hópbílar's profit is decreased. Not every one owns a car. |

Transportation in this solution is provided by employees and Hópbílar.

**Combo 22** : Extreme solution using type 1 and type 7.
**Description** : Solve solution type 1 with as few routes and pickup points as possible. Employees then use a taxi service to said points, type 7.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. This solution is not too simple to be considered an exam project. | A general and special solution that does not take into account employee turnover. Not ALCAN's desired solution. Hard to estimate a general population of an area. Cost might not be decreased. New nodes without a predefined location cannot be used. |
| **External** | Decreases travel time. Employee picked up on doorstep. Increases revenue for taxi service. Service is increased. | Hópbílar's profit is decreased. |

Transportation in this solution is provided by taxi service and Hópbílar.

**Combo 23** : Extreme solution using type 2 and type 7.
**Description** : Solve solution type 2 with as few routes and pickup points as possible. Employees then use a taxi service to said points, type 7.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. This solution is not too simple to be considered an exam project. | A special solution that does not take into account employee turnover. Not ALCAN's desired solution. Cost might not be decreased. New nodes without a predefined location cannot be used. |
| **External** | Decreases travel time. Employee picked up on doorstep. Increases revenue for taxi service. Service is increased. | Hópbílar's profit is decreased. |

Transportation in this solution is provided by taxi service and Hópbílar.

**Combo 24** : Extreme solution using type 3 and type 7.
**Description** : Solve solution type 3 with as few routes and pickup points as possible. Employees then use a taxi service to said points, type 7.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round, 24 hours a day. A special solution takes into account employee turnover. This solution is not too simple to be considered an exam project. | Depends on a third party program. Not ALCAN's desired solution. Hard to estimate a general population of an area. Cost might not be decreased. New nodes without a predefined location cannot be used. |
| **External** | Decreases travel time. Employee picked up on doorstep. Increases revenue for taxi service. Service is increased. Increased profit for the provider of the new soft wear. | Hópbílar's profit is decreased. |

Transportation in this solution is provided by taxi service and Hópbílar.

**Combo 25** : Extreme solution using type 8 and type 7.
**Description** : Solve solution type 8 with as few routes and pickup points as possible. Employees then use a taxi service to said points, type 7.

| | **Helpful** to achieving the objective | **Harmful** to achieving the objective |
|---|---|---|
| **Internal** | Decreases travel time. Works all year round. 24 hours a day. This solution is not too simple to be considered an exam project. | A general and special solution that does not take into account employee turnover. Not ALCAN's desired solution. Hard to estimate a general population of an area. Cost might not be decreased. |
| **External** | Decreases travel time. Employee picked up on doorstep. Increases revenue for taxi service. Service is increased. | Hópbílar's profit is decreased. |

Transportation in this solution is provided by taxi service and Hópbílar.

# Appendix C

# Algorithm

## C.1 Number of Possible Solutions

Consider a set of nodes $V$, the number of nodes in $V$ is n, $|V| = n$. The source node is $j \in V$. It is known for TSP that the number of possible solutions is:

$$(n-1)! \tag{C.1.1}$$

If we relax relax constraint saying all points must be visited and let $i$ denote the number of nodes visited in a certain solution. All nodes are used except the source, since source to source moves are not allowed, although source to sink route is allowed. Now the possible values for $i$ are $i \in \{1, 2, ..., n-1\} = S$.

Next a number of sets, $I_i$, are defined where $I_i \subseteq V, \forall i \in S$. So each set $I_i$ is a reduced version of $N$ that includes the source and $i$ nodes we i.e. have $|I_i| = i+1$. Next $C.1.1$ is applied to each $I_i$ and all possibilities added up:

$$\sum_{i=1}^{n-1}(|I_i|-1)! = \sum_{i=1}^{n-1} i! \tag{C.1.2}$$

This sums up the possibilities for $|S|$ TSP. Each of the $|S|$ TSP only uses $i$ of the $n-1$ available points.For each TSP, other than $i = n-1$,there are more than one posssibility of choosing the $i$ nodes used in the TSP. The possible combination for choice of $i$can be expressed with the binomial ceefficient.

$$C(i) = \binom{n-1}{i} = \frac{(n-1)!}{i!((n-1-i))!} \tag{C.1.3}$$

We now multiply $C.1.2$ and $C.1.3$. This gives the number of possible solutions for a TSP where you have $n$ nodes to choose from but are not restricted to use all, but have to use the source.

$$\sum_{i=1}^{n-1} C(i)(|I_i|-1)! = \sum_{i=1}^{n-1} \frac{(n-1)!}{(n-1-i)!} \tag{C.1.4}$$

If the distance matrix is symmetric then the number of possible solutions is

$$(n-1)+\frac{1}{2}\sum_{i=2}^{n-1}\frac{(n-1)!}{(n-1-i)!} \tag{C.1.5}$$

Because when dealing with 2 points the solution is the same even if the distance matrix is asymmetric. Therefore the solutions dealing with two points are symmetric and therefore do not need to be divided by 2. Now the equation $C.1.7$ can also be written as

$$|S|+\frac{1}{2}\sum_{i=1}^{n-1}\frac{|S|!}{(|S|-i)!} \tag{C.1.6}$$

Where $S=\{1,2,...,n-1\}$ and $|S|=n-1$. This applies for all problems where $n>1$.

Now let us assume that there are $|K|$ routes and that each route, $k$, includes all nodes in the set $V_k$ and the source node, $j$, is in that set, $j \in V_k$. Now we know that $k \in K=\{1,2,...,N\}$, let us now define $V_0=j$. Next we define $J_k$ as the set of all nodes not visited by routes in $k^*$, where $k^*=\{1,2,...,k-1\}$, and the source node. So $J_k=\{V \setminus \{V_0 \cup V_1 ... \cup V_{k-1}, j\}$. It is known that the complexity for each route decreases as a previous route has included some nodes. Then the number of possibilities for a multiple route problem becomes:

$$\sum_{k=0}^{N-1}\sum_{i\in J_k}\frac{(n-1)!}{(n-1-i)!} \tag{C.1.7}$$

## C.2 Algorithm

### C.2.1 Run.java

```
import java.util.Vector;
import java.util.Random;
import java.io.*;
import java.util.Date;
import java.text.DecimalFormat;

public class Sudo10
{

//int K,V;
int V1;
int[][] y;
int[][] Route;
int[] NN;
int OPT;
//File Pro = new File("profit_3_50_a.txt");
//File Tim = new File("dist_3_50_a.txt");
//File Sto = new File("Stop_47.txt");
int[] Profit; //profit matrix
double[][] time;
```

```
int[] Stop;
long date1;
long date2;
long date;
//String file  = "3_50_a_p3test.txt" ;
//String File  = "route3_50_a_p3test.txt" ;
String file1 = "test50a_10_p.txt";
String file2 = "test50a_20_p.txt";
String file3 = "test50a_40_p.txt";
String file4 = "test50a_50_p.txt";
String file5 = "test50a_70_p.txt";
String file6 = "test50a_80_p.txt";
String file7 = "test50a_100_p.txt";
String file8 = "test50a_130_p.txt";
String file9 = "test50a_160_p.txt";

double Time;

public Sudo10(int K, int V, File Pro, File Tim, String File, String file, int[][] p) throws
{
//date1= System.currentTimeMillis();
//K=3;
//V=47;
Profit = new int[V];
time =  new double[V][V];
Stop = new int[V];
NN= new int[9];


NN[0]=10;
NN[1]=20;
NN[2]=40;
NN[3]=50;
NN[4]=70;
NN[5]=80;
NN[6]=100;
NN[7]=130;
NN[8]=160;

FileWriter fw = new FileWriter(file);
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter outFile = new PrintWriter(bw, true);

FileWriter fw2 = new FileWriter(File);
BufferedWriter bw2 = new BufferedWriter(fw2);
PrintWriter outFile2 = new PrintWriter(bw2, true);
```

```
SimulatedAnnealing7 Sim[] = new SimulatedAnnealing7[10];

GetDataFrom1D ProData = new GetDataFrom1D(Profit,Pro);
Profit=ProData.P;

GetDataFrom2D TimeData = new GetDataFrom2D(time, V, Tim);
time=TimeData.P;

//GetDataFrom1D StoData = new GetDataFrom1D(Stop,Sto);
//Stop=StoData.P;

for (int i=0; i<V; i++)
{
if (i==0 || i==V-1)
{
Stop[i]=0;
}
else
{
Stop[i]=1;
}
}

for (int k=0; k<1;k++)
{
for (int i=0;i<9;i++)
{

Decrease DD= new Decrease(V, Profit, time, Stop, NN[i]);
V1=DD.v;

double[][] TT= new double[V1][V1];
TT=DD.T;
int[] profit=new int[V1];
profit=DD.P;
int[] stop =new int[V1];
stop=DD.S;

InitialGuess init = new InitialGuess (K, V1);
Route=init.Route; //constructs the Route matrix 2D

for(int j=0;j<9;j++)
{
date1= System.currentTimeMillis();
Sim[j] = new SimulatedAnnealing7(Route,K, V1, profit, TT, stop, file1, NN[i], p);
OPT=Sim[j].S;
Route=Sim[j].Route;
```

```
date2= System.currentTimeMillis();
date=date2-date1;
Time=Sim[j].TotalT;
outFile.println(date+" "+NN[i]+" "+OPT+" "+Time);
}
}
}
}
}
```

### C.2.2 SimulatedAnnealing.java

```
import java.util.Vector;
import java.util.Random;
import java.lang.Object;
import java.io.*;

public class SimulatedAnnealing7
{
int[][] Route;
int S;  //Solution
int count, MAX=50000; //Maximum number of iterations
//int MaxTime=5; //maximum travel time in a single route
int MRT=0;
int AS=0;
int JCVD=0;
int[] SS;
int iter1=0;
int iter2=0;

//String file = "test_3_50_a_5_1.txt";

double T=15; //temperature
double r=1-Math.pow(10,-13); //reduction factor
double Frozen =2;
double[] SumTime; //time traveled for bus k
double Delta, Delta2;
double prop;
double Num1;
double[] SumTemp;
double TimeTemp;
double Time;
double Maximum;
double TotalT;

float RandN;
```

```
//---------------------------------------
int Snew; //temporary variables
int[][] RouteNew;


calculateOpt calOpt[] = new calculateOpt[2];
calculateTime calT[] = new calculateTime[2];
moves2 move[] = new moves2[2];

public SimulatedAnnealing7(int[][] route, int K, int V, int[] p, double[][] time, int[] Sto
{
Random generator = new Random();

Route=route;
count=0;

//-------------------------------------------------
calOpt[0] = new calculateOpt(Route, K,V, p);

S=calOpt[0].OPT;

//-------------------------------------------------
calT[0]= new calculateTime(Route,time,K,V, Stop);
SumTime=calT[0].SumT;

count=1;


try
{
//-------------------------------------------------
FileWriter fw = new FileWriter(file);
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter outFile = new PrintWriter(bw, true);

while(count<MAX)
{
move[1]=new moves2(K,V,Route, SumTime, MaxTime, time, p, PROP);
RouteNew=move[1].route;

//-------------------------------------------------
calOpt[1] = new calculateOpt(RouteNew, K,V,p);
Snew=calOpt[1].OPT;

//-------------------------------------------------
```

```
Delta=Snew-S;
calculateTime calTemp = new calculateTime(RouteNew, time, K, V, Stop);
SumTemp=calTemp.SumT;
Maximum =0;
TimeTemp=0;
Time=0;

MRT=move[1].temp;
outFile.println("Move: "+MRT);
AS=move[1].temp2;
outFile.println("Instance: "+AS);
JCVD=move[1].temp3;
outFile.println("UsedBuses: "+JCVD);
SS=move[1].UsedBuses;

for(int i=0; i<K; i++)
{
if (Maximum<= SumTemp[i])
{
Maximum=SumTemp[i];
}
TimeTemp=SumTemp[i]+TimeTemp;
Time=SumTime[i]+ Time;
}

if (Maximum <= MaxTime)
{
if (Delta>=0)
{
if (Delta==0)
{
if(TimeTemp<=Time || (Time==0 && TimeTemp!=0))
{
S=calOpt[1].OPT;
Route=move[1].route;
}
else
{
Delta2=TimeTemp-Time;
Num1=-Delta/T;
prop= Math.exp(Num1);
RandN=generator.nextFloat();
if(Delta2 <0 && prop<=RandN && Maximum<= MaxTime)
{
S=calOpt[1].OPT;
Route=move[1].route;
```

```
}
}

//---------------------------
if (TimeTemp<Time || (Time==0 && TimeTemp!=0))
{
iter2=count; //finding the iteration when the optimal value is found
}
//---------------------------

}
else
{
iter1=count; //finding the iteration witch returns the optimal value
S=calOpt[1].OPT;
Route=move[1].route;
}
}
else
{
Num1=-Delta/T;
prop= Math.exp(Num1);
RandN=generator.nextFloat();
if(Delta <0 && prop<=RandN && Maximum<= MaxTime)
{
S=calOpt[1].OPT;
Route=move[1].route;
}
}
}

if (T> Frozen)
{
T=r*T; //Cooling Schedule
}
else
{
T=0;  //Now we implement a local search
}

calT[1]= new calculateTime(Route,time,K,V, Stop);
SumTime=calT[1].SumT;
count=count+1;

//-----------------------------------------------------

try
```

```
{
//outFile.println(S+" " + (count-1));
//outFile.println("iteration:" + (count-1));
//outFile.println("Opt: "+iter1+"    Time: " +iter2);


//for(int k=0; k<K; k++)
//{
//for(int i=0; i<V; i++)
//{
//if(i==0 || Route[k][i]!=0)
//outFile.print(Route[k][i] +" ");
//}
//outFile.println();
//}

TotalT=0;
for(int k=0; k<K; k++)
{
//outFile.print(SumTime[k]+" ");
TotalT=SumTime[k]+TotalT;
}

//outFile.println();
}
catch (NumberFormatException exception)
{
System.out.println ("NumberFormatException" );
}
}

}

catch (IOException exception)
{
System.out.println("IOException ");
}
}
}
```

### C.2.3 moves.java

```
import java.util.Random;
import java.io.*;

public class moves2
{
```

```
int  OPT;
int[] UsedBuses; //number of buses in route
int[] U; // number of unvisited points
int[][] route;
int[] Prop;
int RandProp;
int[][] Rnew;
int Neighbor=6; //number of possible Swaps
int temp=0;
int temp2=0;
int temp3=0;


double SumTravelNew=0;
double SumTravel=0;
double[] Travel;
double[] TravelNew;

calculateTime calT[] = new calculateTime[2];

public moves2 (int K, int V, int[][] Route, double[] SumT, int MaxT, double [][] time, int[
{
Random generator = new Random();
Prop = new int[Neighbor];
route = new int[K][V];

UnvisitedPoints Unvi = new UnvisitedPoints(K,V,Route);
U=Unvi.U;

NumberOfBuses Num = new NumberOfBuses(K,V,Route);
UsedBuses=Num.N;
temp3=UsedBuses.length;

//-----------------------------------------------------------------
//If loop construct the odds of insert or Swap moves happening
//-----------------------------------------------------------------
if (UsedBuses.length>1)//more the one route
{
if (U.length<1 && UsedBuses.length<K) //more then 1 route and no unused points and not all
{ temp2=1;
Prop[0]=PROP[0][0]; //SwapMove2_1 40%
Prop[1]=PROP[0][1]; //SwapMove1_1 40%
Prop[2]=0; //a chosen insert move 0%
Prop[3]=0; //a chosen bus move 0%
Prop[4]=PROP[0][4]; //SwapMove3_1 20%
Prop[5]=0; //Insert1_3 10%
}
```

```
else
{
if (UsedBuses.length==K) //all buses in use
{
if (U.length>0) //unused points available
{ temp2=21;
Prop[0]=PROP[1][0]; //SwapMove2_1 30%
Prop[1]=PROP[1][1]; //SwapMove1_1 30%
Prop[2]=PROP[1][2]; //a chosen insert move 20%
Prop[3]=0; //a chosen bus move 0%
Prop[4]=PROP[1][4]; //SwapMove3_1 10%
Prop[5]=PROP[1][5]; //Insert1_3 10%
}
else  //U.length <=0
{ temp2=22;
Prop[0]=PROP[2][0]; //SwapMove2_1 40%
Prop[1]=PROP[2][1]; //SwapMove1_1 40%
Prop[2]=0; //a chosen insert move 0%
Prop[3]=0; //a chosen bus move 0%
Prop[4]=PROP[2][4]; //SwapMove3_1 20%
Prop[5]=0; //Insert1_3 0%
}
}
else //not all buses in use
{
if (U.length>1) //unused points left
{   temp2=3;
Prop[0]=PROP[3][0]; //SwapMove2_1 30%
Prop[1]=PROP[3][1]; //SwapMove1_1 20%
Prop[2]=PROP[3][2]; //a chosen insert move 25%
Prop[3]=PROP[3][3]; //a chosen bus move 10%
Prop[4]=PROP[3][4]; //SwapMove1_1 10%
Prop[5]=PROP[3][5]; //Insert1_3 5%
}
}

}
}
else
{
if (UsedBuses.length>0) //only one route
{
if(UsedBuses.length>0 && U.length<1) //only one route and no unused points
{   temp2=5;
Prop[0]=0; //SwapMove2_1 0%
Prop[1]=100; //SwapMove1_1 100%
Prop[2]=0; //a chosen insert move 0%
```

```
Prop[3]=0;  //a chosen bus move 0%
Prop[4]=0; //SwapMove3_1 0%
Prop[5]=0; //Insert1_3 0%
}
else //only one route and unused points
{     temp2=6;
Prop[0]=0;  //SwapMove2_1 0%
Prop[1]=PROP[4][1];  //SwapMove1_1 30%
Prop[2]=PROP[4][2];  //a chosen insert move 50%
Prop[3]=PROP[4][3];  //a chosen bus move 10%
Prop[4]=0; //SwapMove3_1 0%
Prop[5]=PROP[4][5]; //Insert1_3 10%
}
}
else  //no route
{    temp2=7;
Prop[0]=0; //SwapMove2_1 0%
Prop[1]=0;  //SwapMove1_1 0%
Prop[2]=PROP[5][2];  //a chosen insert move 90%
Prop[3]=PROP[5][3]; //a chosen bus move 10%
Prop[4]=0; //SwapMove3_1 0%
Prop[5]=0; //Insert1_3 0%
}
}

//-------------------------------------------------------------------

RandProp=Math.abs (generator.nextInt())%(100)+1;//generates an number between 1 & 100

if (RandProp<Prop[0])
{ temp=1;
swapmove2_1 swa= new swapmove2_1(V,K, UsedBuses, Route);
route=swa.r;
}
else//-------------------------------------------------------------
{
if(RandProp<Prop[1]+Prop[0] && RandProp>=Prop[0] && Prop[1]!=0)
{ temp=2;
swapmove1_1 swa= new swapmove1_1(V,K, UsedBuses, Route);
route=swa.r;
}
else//-----------------------------------------------------------------------
{
if(RandProp<Prop[2]+Prop[1]+Prop[0] && RandProp>=Prop[1]+Prop[0] && Prop[2]!=0)
{ temp=3;
InsertMove1_1 ins = new InsertMove1_1(V,K,U,UsedBuses,Route, time, MaxT );
//InsertMove1_2 ins = new InsertMove1_2(V,K,U,UsedBuses,Route, time, MaxT , profit);
```

```
//InsertMove1_4 ins = new InsertMove1_4(V,K,U,UsedBuses,Route, time, MaxT , profit);
//InsertMove1_5 ins = new InsertMove1_5(V,K,U,UsedBuses,Route, time, MaxT , profit);
route=ins.r;
}
else
{
if (RandProp<Prop[3]+Prop[2]+Prop[1]+Prop[0] && RandProp >= Prop[2]+Prop[1]+Prop[0] && Prop
{ temp=4;
BusMove1_1 bus =new BusMove1_1(V,K,U,UsedBuses, Route,time, MaxT);
route=bus.r;
}
else
{
if (RandProp<Prop[4]+ Prop[3]+Prop[2]+Prop[1]+Prop[0] && RandProp >= Prop[3]+ Prop[2]+Prop[
{
temp=5;
swapmove3_1 swa= new swapmove3_1(V,K,UsedBuses,Route);
route=swa.r;
}
else
{
if (Prop[5]!=0)
{
temp=6;
InsertMove1_3 ins = new InsertMove1_3(V,K,U,UsedBuses,Route, profit);
route=ins.r;
}
}
}
}
}
}

}
}
```

### C.2.4 InitialGuess.java

```
import java.util.Vector;
import java.util.Random;

public class InitialGuess
{
int [][] Route;
public InitialGuess(int K, int V)
{
```

```
Route = new int [K][V]; //
for(int k=0; k<=(K-1); k++)
{
Route[k][0]=0;
Route[k][1]=V-1; //V=L+2 but the number of nodes is V-1
}
}
}
```

### C.2.5 calculateOpt.java

```
import java.util.Random;
import java.io.*;

public class calculateOpt
{
int alpha;
int beta;
int sumX;
int sumY;
int OPT;
int R;

public calculateOpt (int[][] route, int K, int V, int[] P)
{
alpha=1;
beta=15;
sumX=0;
sumY=0;
for (int k=0;k<K; k++)
{
for (int i=0;i<V;i++)
{
sumY=sumY+P[route[k][i]];
}
if(route[k][1]==(V-1))
{
sumX=sumX+1;
}
}
OPT=alpha*sumY+beta*sumX;
}
}
```

### C.2.6 calculateTime.java

```
import java.util.Random;
import java.io.*;

public class calculateOpt
{
int alpha;
int beta;
int sumX;
int sumY;
int OPT;
int R;

public calculateOpt (int[][] route, int K, int V, int[] P)
{
alpha=1;
beta=15;
sumX=0;
sumY=0;
for (int k=0;k<K; k++)
{
for (int i=0;i<V;i++)
{
sumY=sumY+P[route[k][i]];
}
if(route[k][1]==(V-1))
{
sumX=sumX+1;
}
}
OPT=alpha*sumY+beta*sumX;
}
}
```

### C.2.7 UnvisitedPoints.java

```
import java.util.Vector;
import java.util.Random;

public class UnvisitedPoints2
{
int[] U;
int sum;
int temp=0;
int temp2=0;
public UnvisitedPoints2(int K, int V, int[][] route, double[][] dist, int MaxDist)
```

```
{
U = new int[0];
sum=0;

for (int i=1; i<V-1; i++)
{
temp=0;
for (int k=0; k<K; k++)
{
for(int j=0; j<V; j++)
{
if (route[k][j]==i)
{
temp=1;
}
}
}
if(temp==0)
{
temp2=0;
for (int k=0; k<K;k++)
{
for (int j=1;j<V; j++)
{
if (dist[route[k][j]][i]< MaxDist && route[k][j]!=i && route[k][j]!=0)
{
temp2=1;
}
}
if (temp2==0)
{
U = addArrayElement(i);
}
}
}
}
public int[] addArrayElement(int n){
int[] newarray = new int[U.length + 1];
for (int i = 0;i < U.length;i++){
newarray[i] = U[i];
}
newarray[U.length] = n;
return newarray;
}

}
```

### C.2.8 NumberOfBuses.java

```
import java.util.Vector;
import java.util.Random;

public class NumberOfBuses
{
int[] N;
public NumberOfBuses(int K, int V, int[][] route)
{
N = new int[0];
//This double for loop finds all buses that are on route
for(int k=0; k<K; k++)
{
if (route[k][1]<V-1)
{
N = addArrayElement(k);
}
}
}
public int[] addArrayElement(int n){
int[] newarray = new int[N.length + 1];
for (int i = 0;i < N.length;i++){
newarray[i] = N[i];
}
newarray[N.length] = n;
return newarray;
}
}
```

### C.2.9 InsertMove11.java

```
import java.util.Vector;
import java.util.Random;
//-------------------------------------------------------------
//Inserts a random node into a random route, currently in use.
//If the route will then become to long a new route (and node)
//will be chosen. The current route will though be tested MaxTemp
//times before it is abandond (it is possible to ad a node without
//increasing travel time)
//-------------------------------------------------------------

public class InsertMove1_1
{
int LengthU;
int RandI;
int LengthK;
```

```
int RandK;
int RandAdd; //the spot where the new node is added into the route
int[][] r; //route
int NewNode; //the node to be added
int AddedTo; //the route to be increased
int NumVisitedPoints; //Number of nodes in chosen route (includes source and sink)
int MAX=100;  //maximum number of iterations
double[] SumT;//travel time for busses 0... K-1


public InsertMove1_1(int V, int K, int[] u, int[] UsedBuses, int[][] Route, double[][] time
{
Random generator = new Random();
r = new int[K][V];
//For loop necessary else epsilon and Y will follow each other.
for (int k=0; k<K; k++)
{
for (int i=0; i<V; i++)
{
r[k][i]=Route[k][i];
}
}
//---------------------------------------------------------------------
//Finding a random node to be inserted
//---------------------------------------------------------------------
LengthU = u.length;
RandI=Math.abs (generator.nextInt())%(LengthU); // Generates a random node
NewNode= u[RandI];
//---------------------------------------------------------------------
//Finding the route the node will be inserted into
//---------------------------------------------------------------------
LengthK= UsedBuses.length;

if (LengthK==0) //no bus in use or current routes are full
{
RandK=Math.abs (generator.nextInt())%(K);
AddedTo=RandK;
}
else
{
RandK=Math.abs (generator.nextInt())%(LengthK); //Generates a bus that is on route
AddedTo=UsedBuses[RandK];
}
//---------------------------------------------------------------------
//Inserting the new node into the route
//---------------------------------------------------------------------
for (int i=0; i<V; i++)
```

```
{
if (r[AddedTo][i]==V-1)
{
NumVisitedPoints=i;// find the number of nodes in the current route
}
}
//System.out.println(NumVisitedPoints);
//System.out.println(AddedTo);

RandAdd=Math.abs (generator.nextInt())%(NumVisitedPoints)+1; // Generates a random number s

//adding the new node into the current route
if (NumVisitedPoints==RandAdd)
r[AddedTo][RandAdd+1]=r[AddedTo][RandAdd];
else{
for (int i=NumVisitedPoints; i>=RandAdd; i--)
{
r[AddedTo][i+1]=r[AddedTo][i];
}}
r[AddedTo][RandAdd]=NewNode;
}
}
```

### C.2.10 BusMove.java

```
import java.util.Vector;
import java.util.Random;
//-------------------------------------------------------------
//Inserts a random nodes into a random route, currently not in use.
//If the route will then become to long a new route (and node)
//will be chosen. The current route will though be tested MaxTemp
//times before it is abandond (it is possible to ad a node without
//increasing travel time)
//-------------------------------------------------------------

public class BusMove1_1
{
int LengthU;
int RandI;
int RandN, RandN2;
int LengthK;
int RandK;
int Num; //number of points to be added to route
int M; //No more the this many points can be in the route
int bool=0; //Boolean number for while loop
int RandAdd; //the spot where the new node is added into the route
```

```
int[] Buses;
int[][] r; //route
int[] NewNodes; //the nodes to be added
int AddedTo; //the route to be increased
int NumVisitedPoints; //Number of nodes in chosen route (includes source and sink)
double SumT;//travel time for the bus route
double T;
int count=0;

public BusMove1_1(int V, int K, int[] u, int[] UsedBuses, int[][] Route, double[][] time, i
{
Random generator = new Random();
r = new int[K][V];
Buses = new int[K];
M=V-1;

//Determinging the length of the new route
LengthU = u.length;
RandI=Math.abs (generator.nextInt())%(LengthU)+1; // Generates a random number
if (RandI>M)
{
Num=M;
}
else
{
Num=RandI;
}
NewNodes =new int[Num];


//For loop necessary else epsilon and Y will follow each other.
for (int k=0; k<K; k++)
{
Buses[k]=k;
for (int i=0; i<V; i++)
{
r[k][i]=Route[k][i];
}
}

//-----------------------------------------------------------------
//Finding the route the nodes will be inserted into
//-----------------------------------------------------------------
LengthK= UsedBuses.length;

if (LengthK==0) //no bus in use or current routes are full
{
```

```
RandK=Math.abs (generator.nextInt())%(K);
AddedTo=RandK;
}
else
{
for (int i=0; i<LengthK; i++)
{
for(int j=0; j<Buses.length; j++)
{
if(Buses[j]==UsedBuses[i])
{
Buses = removeArrayElement(j); //removing used buses from the function
}
}
}
RandK=Math.abs (generator.nextInt())%(Buses.length); //Generates a random bus
AddedTo=Buses[RandK];
}
//---------------------------------------------------------------------
//Finding a random nodes to be inserted & inserting them
//---------------------------------------------------------------------
for (int i=0; i<Num; i++)
{
RandN=Math.abs (generator.nextInt())%(LengthU);
NewNodes[i]= u[RandN];
r[AddedTo][i+1]=NewNodes[i];
UnvisitedPoints Unvi = new UnvisitedPoints(K,V,r); //unused nodes redefined
u=Unvi.U;
LengthU= u.length;
}
//---------------------------------------------------------------------
r[AddedTo][Num+1]=V-1;
}
  /**  Creates a new array from intarray skipping element n.
  */
//For Buses Matrix
public int[] removeArrayElement(int n){
    int[] newarray = new int[Buses.length - 1];
    for (int i = 0;i < Buses.length;i++){
    if (i < n)
    {
newarray[i] = Buses[i];
}

    if (i > n)
    {
newarray[i-1] = Buses[i];
```

```
    }
    }
    return newarray;
    }
// For NewNodes matrix
public int[] removeArrayElement2(int n){
    int[] newarray2 = new int[NewNodes.length - 1];
    for (int i = 0;i < NewNodes.length;i++){
    if (i < n)  newarray2[i] = NewNodes[i];
    if (i > n)  newarray2[i-1] = NewNodes[i];
    }
    return newarray2;
    }
}
```

### C.2.11 SwapMove11.java

```
import java.util.Vector;
import java.util.Random;
//------------------------------------------------------------
//Swaps to nodes in the same route
//Can only be entered if there is an active route in the system.
//A swap move will only decrease traveling time and not profit
//------------------------------------------------------------

public class swapmove1_1
{
int temp;
int RandI;
int LengthK;
int RandK;
int RandSwap1;
int RandSwap2;
int SwappedIn; //the spot where the new node is added into the route
int[][] r; //route
int NumVisitedPoints; //Number of nodes in chosen route (includes source and sink)
int MAX=10;
int count=0;

public swapmove1_1(int V, int K, int[] UsedBuses, int[][] Route)
{
Random generator = new Random();
r= new int[K][V];

for(int k=0; k<K; k++ )
{
```

```
for(int i=0;i<V; i++ )
{
r[k][i]=Route[k][i];
}
}
LengthK= UsedBuses.length;
//This move is only feasible when there already some buses in route, visiting more the one
RandK=Math.abs (generator.nextInt())%(LengthK); //Generates a bus that is on route
SwappedIn=UsedBuses[RandK];

//Finding the number of nodes in route, they have to be at least 2
for (int i=0; i<V; i++)
{
if (r[SwappedIn][i]==V-1)
{
NumVisitedPoints=i;// find the number of nodes in the current route
}
}

RandSwap1=Math.abs (generator.nextInt())%(NumVisitedPoints-1)+1;
RandSwap2=Math.abs (generator.nextInt())%(NumVisitedPoints-1)+1;

while (RandSwap1==RandSwap2 && count<MAX && NumVisitedPoints>2)
{
RandSwap1=Math.abs (generator.nextInt())%(NumVisitedPoints-1)+1; // Generates a random numb
RandSwap2=Math.abs (generator.nextInt())%(NumVisitedPoints-1)+1;
count=count+1;
}
temp=r[SwappedIn][RandSwap1];
r[SwappedIn][RandSwap1]=r[SwappedIn][RandSwap2];
r[SwappedIn][RandSwap2]=temp;
}
}
```

### C.2.12 SwapMove21.java

```
import java.util.Vector;
import java.util.Random;
//-------------------------------------------------------------
//Swaps to nodes in the same route
//Can only be entered if there are 2 our more routes active
//
//-------------------------------------------------------------

public class swapmove2_1
{
```

```
int temp;
int RandI;
int LengthK;
int RandK1;
int RandK2;
int RandSwap1;
int RandSwap2;
int SwappedBetween1; //the spot where the new node is added into the route
int SwappedBetween2;
int[][] r; //route
int NumVisitedPoints1; //Number of nodes in chosen route (includes source and sink)
int NumVisitedPoints2;
public swapmove2_1(int V, int K, int[] UsedBuses, int[][] Route)
{
Random generator = new Random();
r= new int[K][V];

for(int k=0; k<K; k++ )
{
for(int i=0;i<V; i++ )
{
r[k][i]=Route[k][i];
}
}

LengthK= UsedBuses.length;
//This move is only feasible when there already some buses in route, visiting more the one
while (RandK1==RandK2)
{
RandK1=Math.abs (generator.nextInt())%(LengthK); //Generates a bus that is on route
RandK2=Math.abs (generator.nextInt())%(LengthK);
}

SwappedBetween1=UsedBuses[RandK1];
SwappedBetween2=UsedBuses[RandK2];

//Finding the number of nodes in route, they have to be at least 2
for (int i=0; i<V; i++)
{
if (r[SwappedBetween1][i]==V-1)
{
NumVisitedPoints1=i;// find the number of nodes in the current route
}
}
for (int i=0; i<V; i++)
{
if (r[SwappedBetween2][i]==V-1)
```

```
{
NumVisitedPoints2=i;// find the number of nodes in the current route
}
}
RandSwap1=Math.abs (generator.nextInt())%(NumVisitedPoints1-1)+1; // Generates a random num
RandSwap2=Math.abs (generator.nextInt())%(NumVisitedPoints2-1)+1;

temp=r[SwappedBetween1][RandSwap1];
r[SwappedBetween1][RandSwap1]=r[SwappedBetween2][RandSwap2];
r[SwappedBetween2][RandSwap2]=temp;
}
}
```

### C.2.13 SwapMove31.java

```
import java.util.Vector;
import java.util.Random;
//------------------------------------------------------------
//Swaps one nodes from one route to anouther
//Can only be entered if there is more then one active route in the system.
//A swap move will only decrease traveling time and not profit
//------------------------------------------------------------

public class swapmove3_1
{
int temp;
int RandI;
int LengthK;
int RandK1;
int RandK2;
int RandFrom;
int RandTo;
int RandSwap;
int RandLocation;
int SwappedFrom;
int SwappedTo;
int[][] r; //route
int NumVisitedPoints1; //Number of nodes in SwappedFrom route (includes source and sink)
int NumVisitedPoints2; //Number of nodes in SwappedTo route (includes source and sink)
int MAX=50;
int count=0;

public swapmove3_1(int V, int K, int[] UsedBuses, int[][] Route)
{
Random generator = new Random();
r= new int[K][V];
```

```
for(int k=0; k<K; k++ )
{
for(int i=0;i<V; i++ )
{
r[k][i]=Route[k][i];
}
}
LengthK= UsedBuses.length;

//This move is only feasible when there already some buses in route, visiting more the one
RandK1=Math.abs (generator.nextInt())%(LengthK); //Generates a bus that is on route
SwappedFrom=UsedBuses[RandK1];

RandK2=Math.abs (generator.nextInt())%(LengthK); //Generates anouther bus that is on route
SwappedTo=UsedBuses[RandK2];

while (RandK1==RandK2 && count<MAX)
{
RandK1=Math.abs (generator.nextInt())%(LengthK); //Generates a bus that is on route
SwappedFrom=UsedBuses[RandK1];

RandK2=Math.abs (generator.nextInt())%(LengthK); //Generates anouther bus that is on route
SwappedTo=UsedBuses[RandK2];

count=count+1;
}
//Finding the number of nodes in route, they have to be at least 2
for (int i=0; i<V; i++)
{
if (r[SwappedFrom][i]==V-1)
{
NumVisitedPoints1=i;// find the number of nodes in the SwappedFrom route
}

if (r[SwappedTo][i]==V-1)
{
NumVisitedPoints2=i; // find the number of nodes in the SwappedTo route
}
}
RandSwap=Math.abs (generator.nextInt())%(NumVisitedPoints1-1)+1; // Generates a random numb
RandLocation=Math.abs (generator.nextInt())%(NumVisitedPoints2)+1; // Generates a random nu

temp=r[SwappedFrom][RandSwap]; //the node to be moved

//Adding the node to SwappedTo route
//-----------------------------------------------------------
```

```
for (int i=NumVisitedPoints2; i>= RandLocation; i--)
{
r[SwappedTo][i+1]=r[SwappedTo][i];
}

if (RandLocation==NumVisitedPoints2)
{
r[SwappedTo][NumVisitedPoints2+1]=V-1;
}

r[SwappedTo][RandLocation]=temp;
//-------------------------------------------------------

//Removing the node from SwappedFrom
//-------------------------------------------------------
for (int i=RandSwap; i<=(NumVisitedPoints1+1);i++)
{
r[SwappedFrom][i]=r[SwappedFrom][i+1];
}
}
}
```

# Appendix D

# Test

This, the appendix, contains additional information on many of the test, experimants and analysis carried out in the report. In some cases $p$ may represent the probabiltiy matrix instead of $P$. This is due to change lat in the project wher $P$ was denoted as the probability matrix as $p$ could confuse with the profit of a single node.

## D.1 Non-Randomly Generated Data Sets

### D.1.1 Results Data Set 3_50_$b$

Results $P_2$

```
    52.5  105.5      168.1      211.6      279.5      334.7      374.2     400.6     419.9
      55    120        180        250        301        361        411       432       432
 0.95455      0.87917    0.93389     0.8464    0.92857    0.92715    0.91046   0.92731   0
  3888.7 5790.6       7312     9762.7      10273     9784.4     9572.1    9388.6    9127.7
```

Results $P_1$

```
    55   117        171        249      292.9      356.9      401.8     428.9       423
    55   120        180        250        301        361        411       432       432
     1 0.975       0.95      0.996    0.97309    0.98864    0.97762   0.99282   0.97917
4833.6       6275.6     8560.7      10993      12022      11570      11125     10317
```

Results $P_3$

```
    53.5    114      167.1      213.1      279.7      350.8      376.2     404.8     413.7
      55    120        180        250        301        361        411       432       432
 0.97273   0.95    0.92833     0.8524    0.92924    0.97175    0.91533   0.93704   0.95764
  4216.7 5843.1     7815.3     9982.5      10597      10098     9943.8    9878.2    9470.6
```

### D.1.2 Results Data Set 3_50_$c$

Results $P_2$

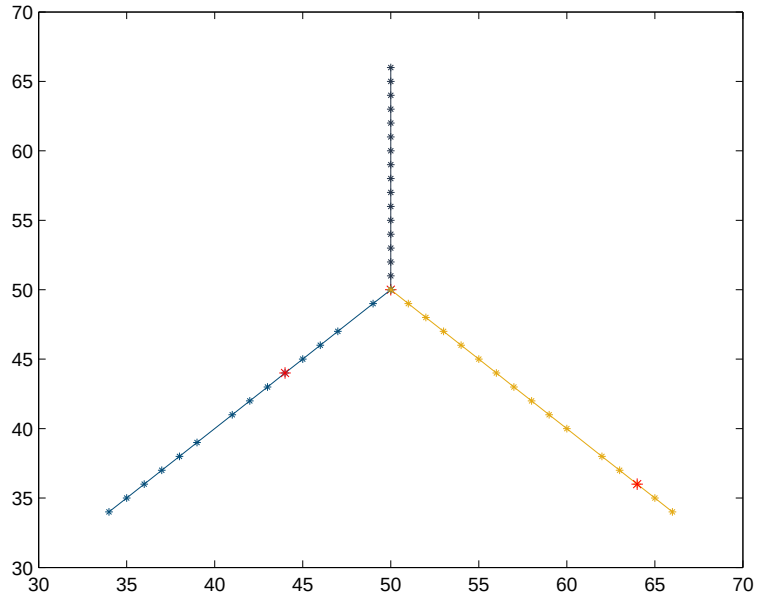



Figure D.1: Shows the data set 3_50_$b$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

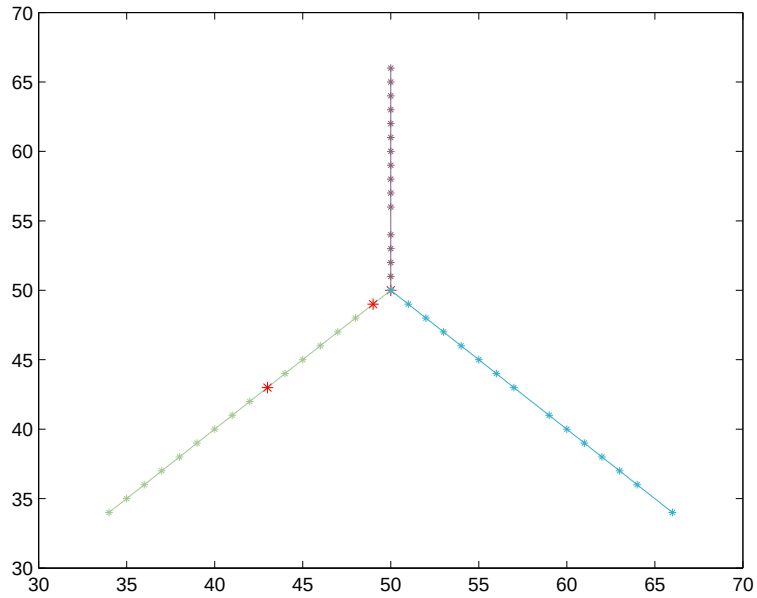



Figure D.2: Shows the data set 3_50_$c$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

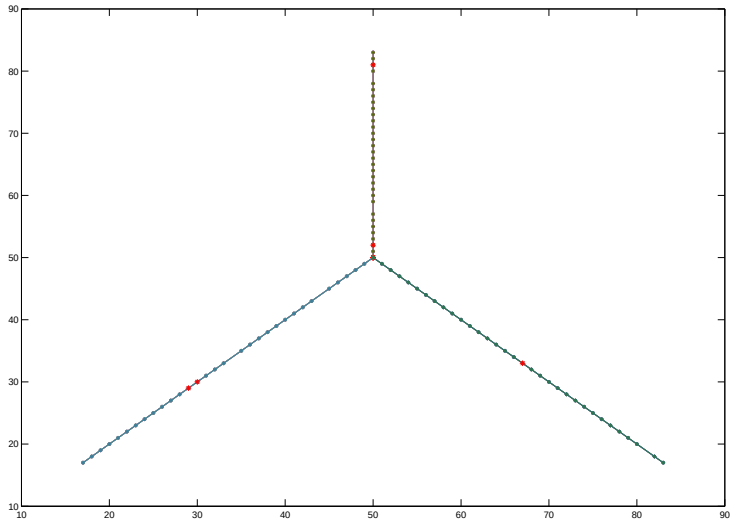



Figure D.3: Shows the data set 3_100_$a$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

```
   51.5     94       166     208.9     275.7   341.9     371.5     398.7     420
     55    111       181       232     300     360       402       432     432
0.93636      0.84685   0.91713   0.90043     0.919   0.94972   0.92413   0.92292   0.9722
 3808.2 5724.8    7530.7     10032     10144  9506.1    9350.6    9352.9    9051.4
```

Results $P_1$

```
   55 104.5       179     228.1     299.2     359   402.9       428     429
   55   111       181       231     300     360     411       432     432
    1      0.94144   0.98895   0.98745   0.99733   0.99722 0.98029   0.99074   0.99306
4843.8       6605.6    8756.9     11517     12029   11760   11730     11412     10319
```

Results $P_3$

```
   55 109.8     171.9     218.4     254.2     341   364.4       410     411
   55   111       181       231     300     360     411       432     432
    1      0.98919   0.94972   0.94545   0.84733   0.94722 0.88662   0.94907   0.95139
4445.5       6130.9    7790.1     10633     10959   10435   10082     10139     10004
```

### D.1.3 Results Data Set 3_100_$a$

Results $P_1$

```
      111.2       216.5       342.2       378.5       447.4       494.4       552.2
        121         240         370         430         490         551         611
    0.91901     0.90208     0.92486     0.88023     0.91306     0.89728     0.90376
     5288.2       10992       21209       25362       29351       30888       32924
```

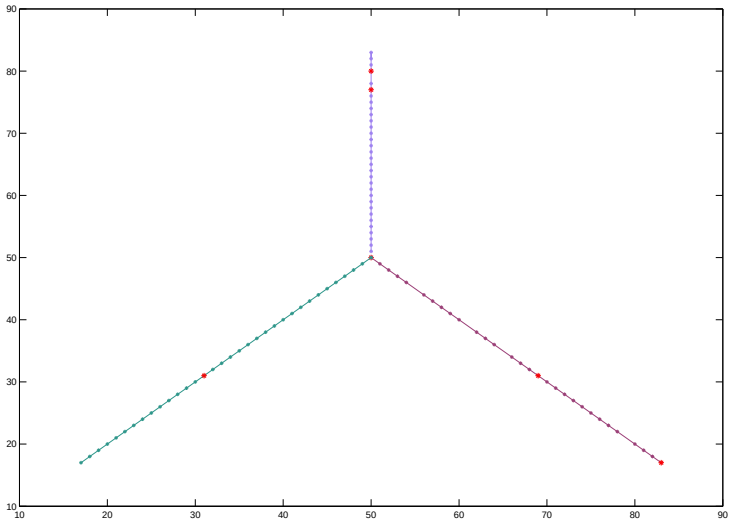



Figure D.4: Shows the data set 3_100_$b$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

Results $P_2$

| | | | | | | |
|---|---|---|---|---|---|---|
| 118 | 231.1 | 358.1 | 414.7 | 492.1 | 521.3 | 587.4 |
| 121 | 240 | 370 | 431 | 500 | 560 | 620 |
| 0.97521 | 0.96292 | 0.96784 | 0.96218 | 0.9842 | 0.93089 | 0.94742 |
| 6121.4 | 12729 | 23002 | 30998 | 34771 | 41077 | 43730 |

Results $P_3$

| | | | | | | |
|---|---|---|---|---|---|---|
| 115.4 | 225.5 | 343.4 | 409.6 | 455.9 | 514.9 | 544. |
| 121 | 240 | 370 | 431 | 491 | 561 | 611 |
| 0.95372 | 0.93958 | 0.92811 | 0.95035 | 0.92851 | 0.91783 | 0.89116 |
| 5431.9 | 11390 | 22847 | 26672 | 33301 | 38870 | 35621 |

### D.1.4 Results Data Set 3_100_$b$

Results $P_1$

| | | | | | | |
|---|---|---|---|---|---|---|
| 113.5 | 219 | 338.5 | 383 | 460 | 467.7 | 563.5 |
| 130 | 240 | 380 | 430 | 500 | 541 | 611 |
| 0.87308 | 0.9125 | 0.89079 | 0.8907 | 0.92 | 0.86451 | 0.92226 |
| 5521.1 | 10912 | 19956 | 29231 | 29985 | 34844 | 37307 |

Results $P_2$

| | | | | | | |
|---|---|---|---|---|---|---|
| 130 | 246 | 359 | 410 | 475 | 540.2 | 594 |

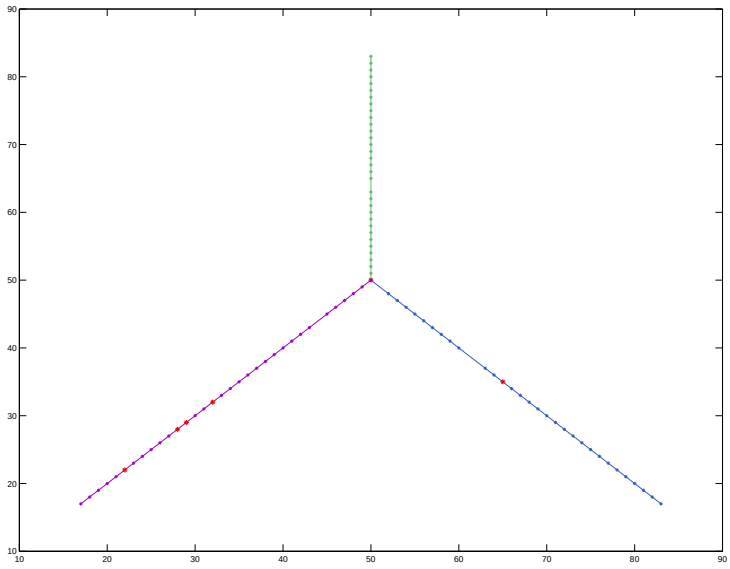



Figure D.5: Shows the data set 3_100_$c$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

| | | | | | | |
|---|---|---|---|---|---|---|
| 130 | 250 | 380 | 430 | 500 | 560 | 620 |
| 1 | 0.984 | 0.94474 | 0.95349 | 0.95 | 0.96464 | 0.95806 |
| 6071.3 | 12023 | 23078 | 28914 | 36920 | 37747 | 44592 |

Results $P_3$

| | | | | | | |
|---|---|---|---|---|---|---|
| 120 | 240 | 354 | 391 | 444.2 | 501.6 | 580.5 |
| 130 | 250 | 380 | 440 | 500 | 541 | 621 |
| 0.92308 | 0.96 | 0.93158 | 0.88864 | 0.8884 | 0.92717 | 0.93478 |
| 5617.7 | 11396 | 22569 | 26318 | 36042 | 40019 | 39442 |

### D.1.5 Results Data Set 3_100_$c$

Results $P_1$

| | | | | | | |
|---|---|---|---|---|---|---|
| 106.5 | 221 | 330 | 395.1 | 425.2 | 445.3 | 526.6 |
| 120 | 240 | 350 | 420 | 471 | 550 | 601 |
| 0.8875 | 0.92083 | 0.94286 | 0.94071 | 0.90276 | 0.80964 | 0.87621 |
| 5301.6 | 10011 | 18860 | 24295 | 30413 | 35980 | 37782 |

Results $P_2$

| | | | | | | |
|---|---|---|---|---|---|---|
| 112 | 237 | 328 | 411 | 456 | 526.3 | 563.5 |
| 120 | 240 | 350 | 420 | 480 | 551 | 601 |
| 0.93333 | 0.9875 | 0.93714 | 0.97857 | 0.95 | 0.95517 | 0.9376 |
| 5802.6 | 11083 | 22702 | 27548 | 35322 | 39322 | 38953 |

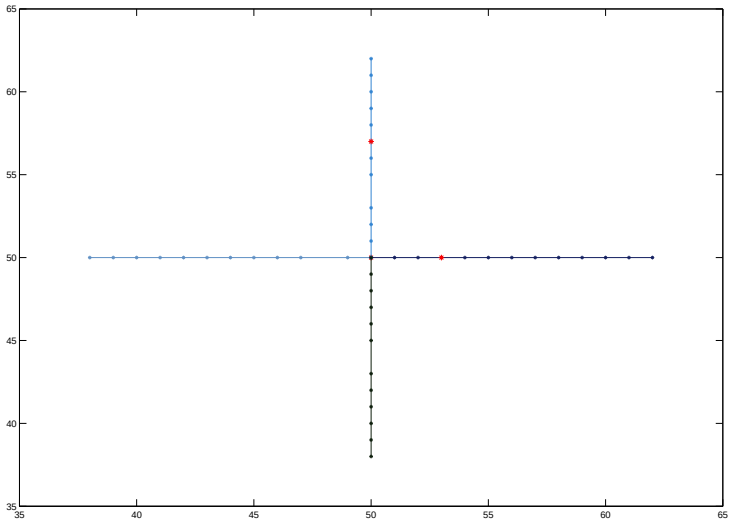



Figure D.6: Shows the data set 4_50_$a$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

Results $P_3$

| | | | | | | |
|---|---|---|---|---|---|---|
| 111 | 223 | 343 | 407 | 455 | 503.7 | 531.1 |
| 120 | 240 | 350 | 420 | 480 | 551 | 591 |
| 0.925 | 0.92917 | 0.98 | 0.96905 | 0.94792 | 0.91416 | 0.89865 |
| 5530.4 | 10180 | 24435 | 26076 | 33234 | 38273 | 42131 |

### D.1.6 Results Data Set 4_50_$a$

Results for $P = P_1$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 62.5 | 109.4 | 174.1 | 262.1 | 342 | 402.7 | 417.9 | 416.8 | |
| 70 | 131 | 201 | 281 | 361 | 432 | 432 | 432 | 43 |
| 0.89286 | 0.83511 | 0.86617 | 0.93274 | 0.94737 | 0.93218 | 0.96736 | 0.96481 | 0 |
| 4790.1 | 8160.9 | 14217 | 12665 | 11754 | 11243 | 11168 | 11130 | |

Results for $P = P_2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 72.5 | 129.6 | 190.8 | 272 | 359.9 | 423 | 431.9 | 430.9 | |
| 75 | 131 | 201 | 281 | 361 | 432 | 432 | 432 | 43 |
| 0.96667 | 0.98931 | 0.94925 | 0.96797 | 0.99695 | 0.97917 | 0.99977 | 0.99745 | 0 |
| 6816.4 | 9613.7 | 17074 | 14932 | 14440 | 13535 | 12780 | 12427 | |

Results for $P = P_3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 70 | 127.6 | 182.8 | 264.5 | 331 | 407.8 | 403.9 | 397.8 | |
| 75 | 131 | 201 | 281 | 361 | 432 | 432 | 432 | 43 |
| 0.93333 | 0.97405 | 0.90945 | 0.94128 | 0.9169 | 0.94398 | 0.93495 | 0.92083 | 0 |
| 5608.3 | 8630.3 | 15744 | 13947 | 12237 | 11754 | 11528 | 11431 | |

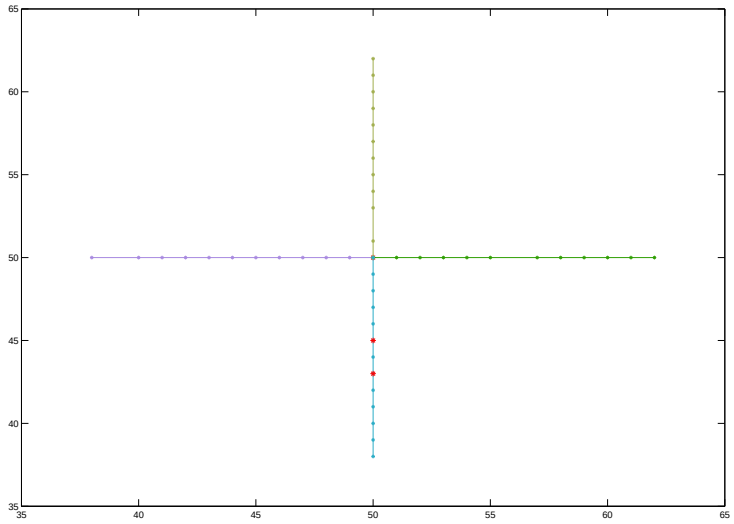



Figure D.7: Shows the data set 4_500_$b$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

### D.1.7 Results Data Set 4_50_$b$

Results for $P = P_1$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 62.5 | 126.5 | 167.4 | 264.7 | 324 | 379 | 399.8 | 402.7 | |
| 70 | 150 | 211 | 290 | 360 | 432 | 432 | 432 | |
| 0.89286 | 0.84333 | 0.79336 | 0.91276 | 0.9 | 0.87731 | 0.92546 | 0.93218 | |
| 4928.6 | 8316.7 | 14414 | 12453 | 11638 | 11345 | 11129 | 11180 | |

Results for $P = P_2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 71.5 | 144 | 206.9 | 273.4 | 364.1 | 423.9 | 423.7 | 417.9 | |
| 75 | 150 | 211 | 290 | 370 | 432 | 432 | 432 | |
| 0.95333 | 0.96 | 0.98057 | 0.94276 | 0.98405 | 0.98125 | 0.98079 | 0.96736 | |
| 7025.3 | 9696.4 | 16331 | 14673 | 14050 | 12864 | 13007 | 12435 | |

Results for $P = P_3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 69.5 | 142 | 205.9 | 275.6 | 333 | 417.8 | 406.8 | 428 | |
| 75 | 150 | 211 | 290 | 362 | 432 | 432 | 432 | |
| 0.92667 | 0.94667 | 0.97583 | 0.95034 | 0.91989 | 0.96713 | 0.94167 | 0.99074 | |
| 6052.4 | 8526.6 | 14891 | 13404 | 11996 | 11668 | 11728 | 11320 | |

### D.1.8 Results Data Set 4_50_$c$

Results for $P = P_1$

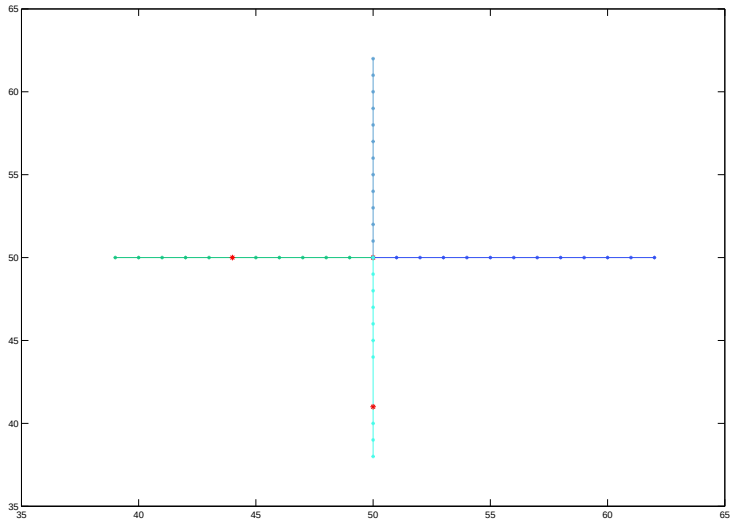



Figure D.8: Shows the data set 4_50_$c$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 64.5 | 146.5 | 191.6 | 271.2 | 343.1 | 408.5 | 406.5 | 424.8 | |
| 70 | 160 | 221 | 292 | 372 | 432 | 432 | 432 | 4 |
| 0.92143 | 0.91563 | 0.86697 | 0.92877 | 0.92231 | 0.9456 | 0.94097 | 0.98333 | 0 |
| 5201.6 | 8204.9 | 14624 | 12352 | 11850 | 11494 | 11322 | 11218 | |

Results for $P = P_2$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 75 | 156 | 229.9 | 286.4 | 361.5 | 423.9 | 428 | 428.9 | |
| 80 | 160 | 231 | 292 | 372 | 432 | 432 | 432 | 4 |
| 0.9375 | 0.975 | 0.99524 | 0.98082 | 0.97177 | 0.98125 | 0.99074 | 0.99282 | |
| 6975.3 | 9060.5 | 15812 | 14074 | 13112 | 12542 | 12264 | 12323 | |

Results for $P = P_3$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 71.5 | 148.5 | 228.9 | 277.5 | 342.1 | 412.3 | 406.5 | 422.8 | |
| 80 | 160 | 231 | 292 | 372 | 432 | 432 | 432 | 4 |
| 0.89375 | 0.92812 | 0.99091 | 0.95034 | 0.91962 | 0.9544 | 0.94097 | 0.9787 | 0 |
| 6096.6 | 8785.3 | 15063 | 12749 | 12204 | 11575 | 11507 | 11309 | |

### D.1.9 Results Data Set 4_100_$a$

Results $P_1$

| | | | | | | |
|---|---|---|---|---|---|---|
| 167.5 | 295.6 | 413.5 | 485.7 | 593.1 | 624.6 | 681.3 |
| 190 | 370 | 471 | 560 | 633 | 712 | 793 |
| 0.88158 | 0.79892 | 0.87792 | 0.86732 | 0.93697 | 0.87725 | 0.85914 |
| 10836 | 39478 | 55322 | 47922 | 45993 | 42580 | 38845 |

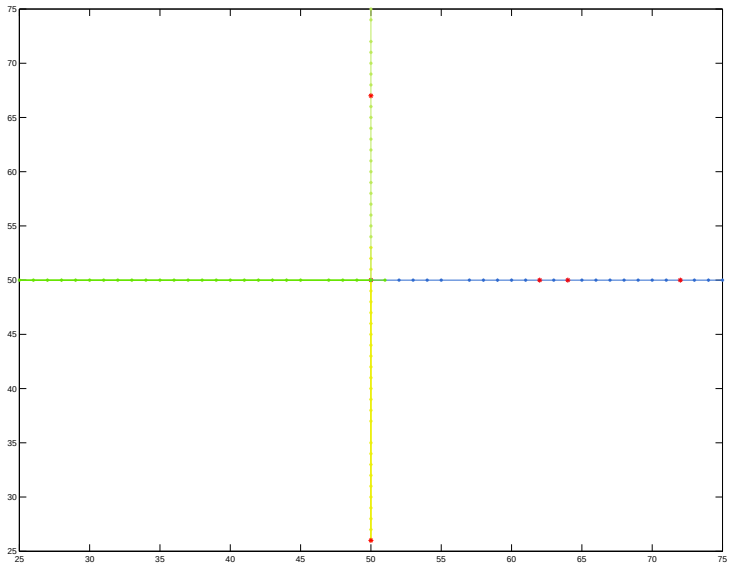



Figure D.9: Shows the data set 4_100_$a$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

Results $P_2$

| | | | | | | |
|---|---|---|---|---|---|---|
| 183 | 362 | 462.6 | 544.8 | 628 | 697.2 | 742.3 |
| 190 | 380 | 471 | 561 | 661 | 750 | 814 |
| 0.96316 | 0.95263 | 0.98217 | 0.97112 | 0.95008 | 0.9296 | 0.91192 |
| 11610 | 44446 | 69766 | 57475 | 53055 | 47969 | 44791 |

Results $P_3$

| | | | | | | |
|---|---|---|---|---|---|---|
| 183 | 351.3 | 423.5 | 527.7 | 577.3 | 631.2 | 672.8 |
| 190 | 380 | 471 | 561 | 660 | 712 | 763 |
| 0.96316 | 0.92447 | 0.89915 | 0.94064 | 0.8747 | 0.88652 | 0.88178 |
| 11258 | 43975 | 67787 | 63596 | 52416 | 49354 | 43542 |

### D.1.10 Results Data Set 4_100_$b$

Results $P_1$

| | | | | | | |
|---|---|---|---|---|---|---|
| 165.5 | 315 | 425.2 | 508 | 597.4 | 664.8 | 645.7 |
| 190 | 360 | 470 | 550 | 641 | 751 | 761 |
| 0.87105 | 0.875 | 0.90468 | 0.92364 | 0.93198 | 0.88522 | 0.84849 |
| 10984 | 35607 | 55929 | 51928 | 47081 | 41341 | 38896 |

Results $P_2$

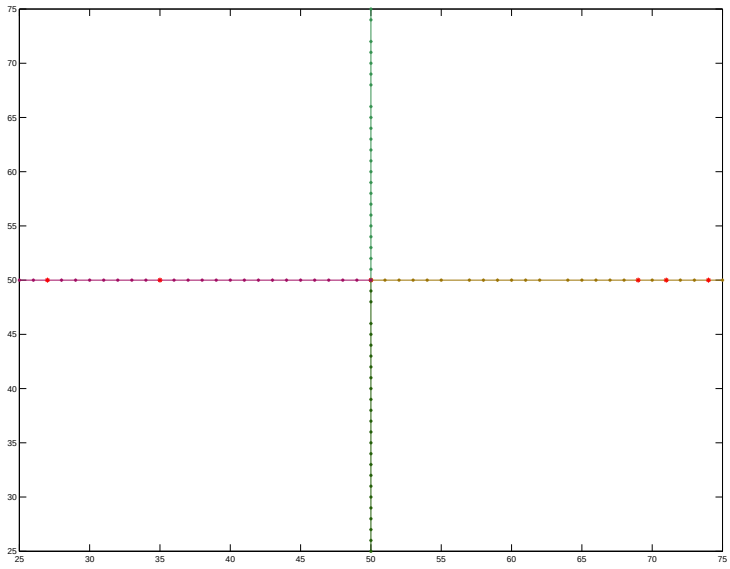



Figure D.10: Shows the data set 4_100_$b$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

| | | | | | | |
|---|---|---|---|---|---|---|
| 182 | 377 | 440 | 566.6 | 651.2 | 684.5 | 790.2 |
| 190 | 380 | 470 | 571 | 670 | 751 | 841 |
| 0.95789 | 0.99211 | 0.93617 | 0.99229 | 0.97194 | 0.91145 | 0.9396 |
| 11757 | 39682 | 65734 | 58835 | 56639 | 47337 | 44403 |

Results $P_3$

| | | | | | | |
|---|---|---|---|---|---|---|
| 182 | 361 | 426 | 532.3 | 625.8 | 668.3 | 688.6 |
| 190 | 380 | 470 | 571 | 661 | 731 | 752 |
| 0.95789 | 0.95 | 0.90638 | 0.93222 | 0.94675 | 0.91423 | 0.91569 |
| 11515 | 44669 | 84627 | 62193 | 51611 | 48193 | 45368 |

### D.1.11 Results Data Set 4_100_$c$

Results $P_1$

| | | | | | | |
|---|---|---|---|---|---|---|
| 157.5 | 341 | 428.5 | 520.4 | 607.4 | 643.7 | 693.8 |
| 190 | 380 | 470 | 570 | 660 | 700 | 780 |
| 0.82895 | 0.89737 | 0.9117 | 0.91298 | 0.9203 | 0.91957 | 0.88949 |
| 11248 | 35945 | 58997 | 54339 | 48585 | 42226 | 40935 |

Results $P_2$

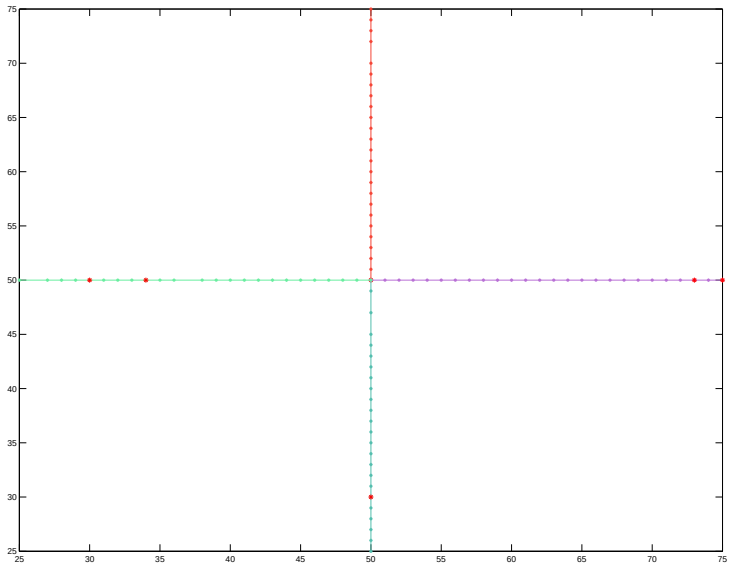



Figure D.11: Shows the data set 4_100_$c$, points in red mark the point with decreased profit or the depot(center). One can also see which point have been removed form the routes.

| | | | | | | |
|---|---|---|---|---|---|---|
| 190 | 378 | 463 | 548 | 644.3 | 681.4 | 793.6 |
| 190 | 390 | 470 | 580 | 661 | 761 | 841 |
| 1 | 0.96923 | 0.98511 | 0.94483 | 0.97474 | 0.8954 | 0.94364 |
| 11702 | 42901 | 69121 | 60838 | 53120 | 47479 | 46224 |

Results $P_3$

| | | | | | | |
|---|---|---|---|---|---|---|
| 180 | 356 | 446 | 551.1 | 596.4 | 668.7 | 721.5 |
| 190 | 390 | 470 | 580 | 661 | 731 | 802 |
| 0.94737 | 0.91282 | 0.94894 | 0.95017 | 0.90227 | 0.91477 | 0.89963 |
| 11513 | 49877 | 70498 | 52944 | 53646 | 46272 | 45828 |

## D.2 Cooling Schedule

### D.2.1 Results for temperature, $T$, first run

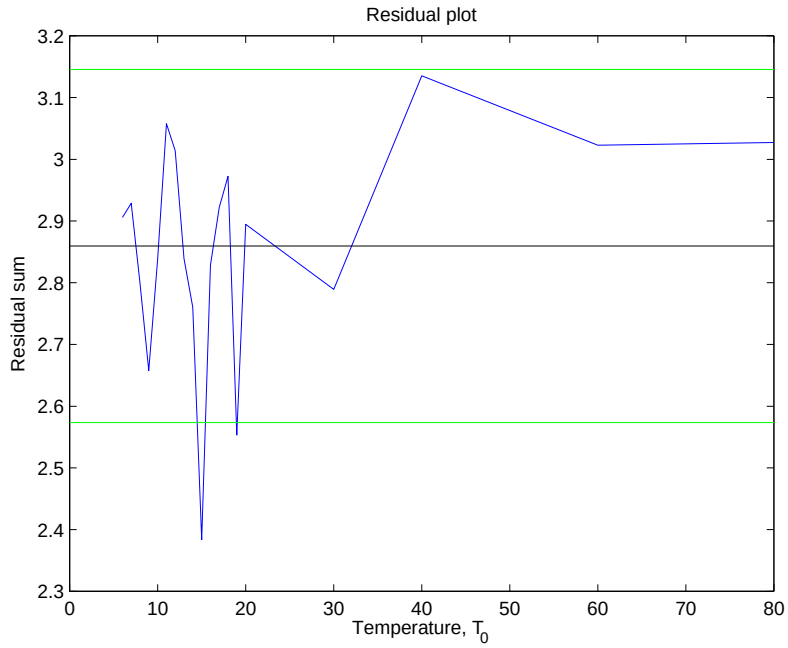



Figure D.12: This figues shows the residual sum for some temperatures, blue line. The black dots are the mean residual sum for all temperatues and green dots are mean±10%.

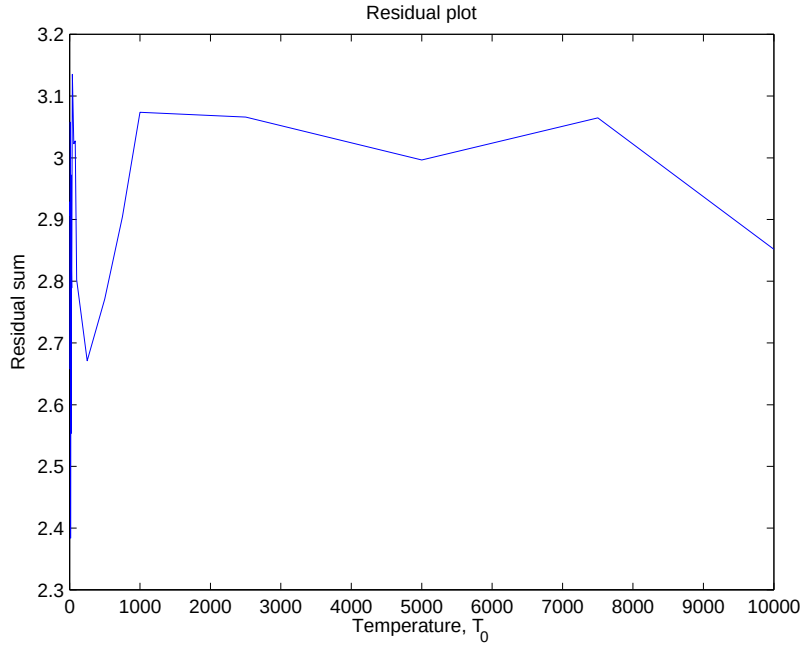



Figure D.13: This figues shows the residual sum for all tested temperatures.

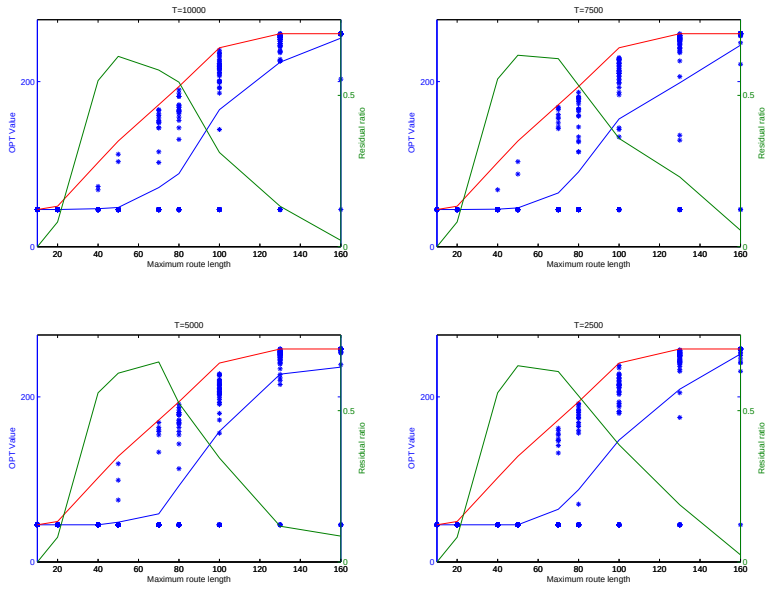



Figure D.14: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

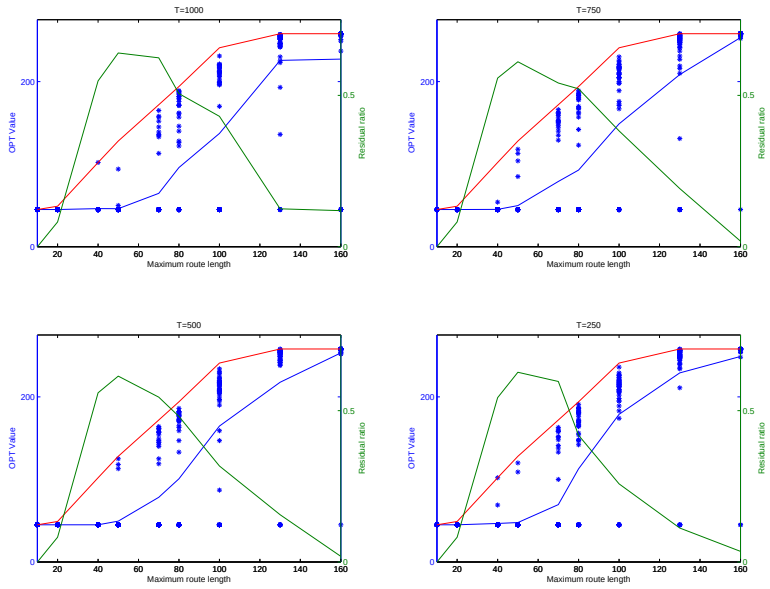



Figure D.15: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

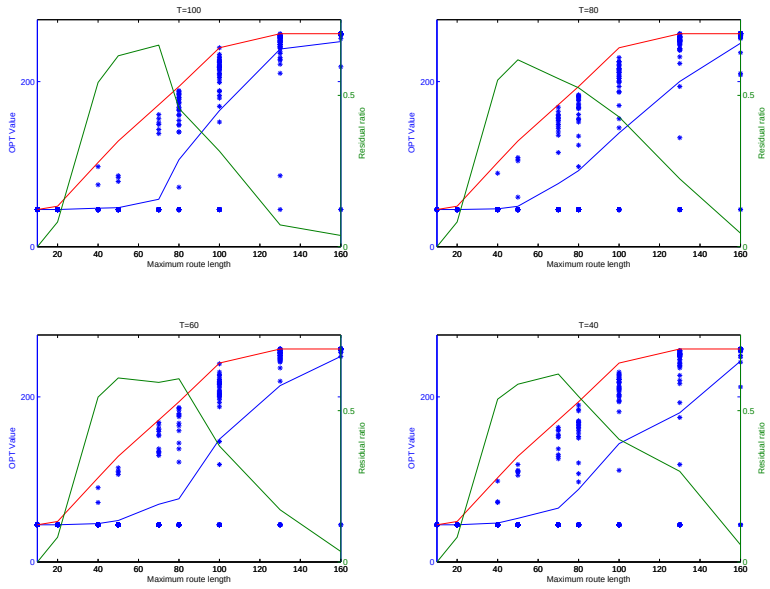



Figure D.16: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

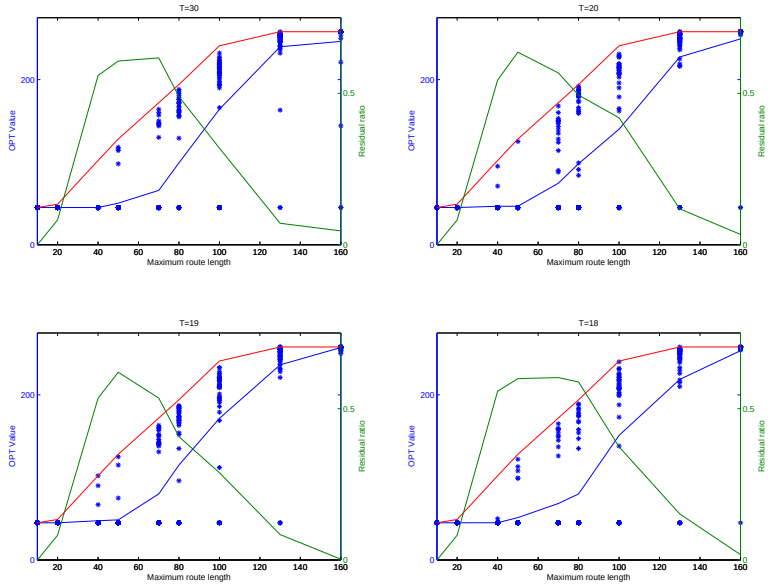



Figure D.17: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

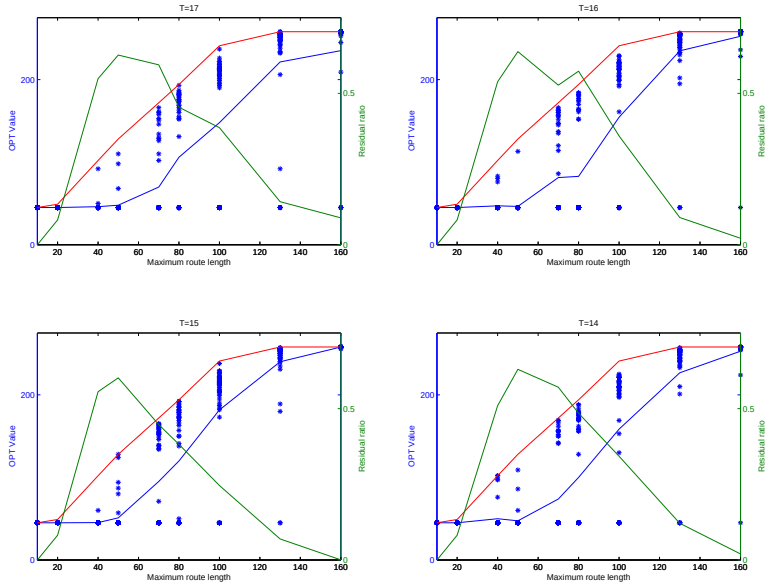



Figure D.18: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

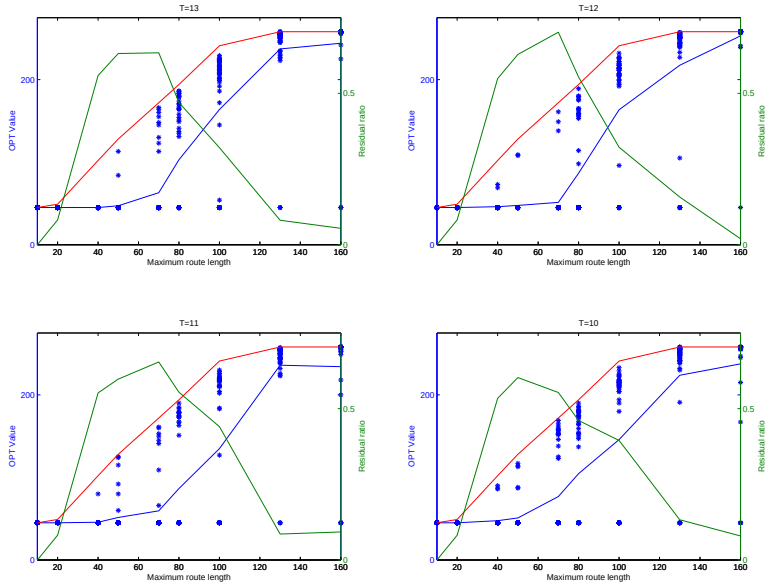



Figure D.19: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

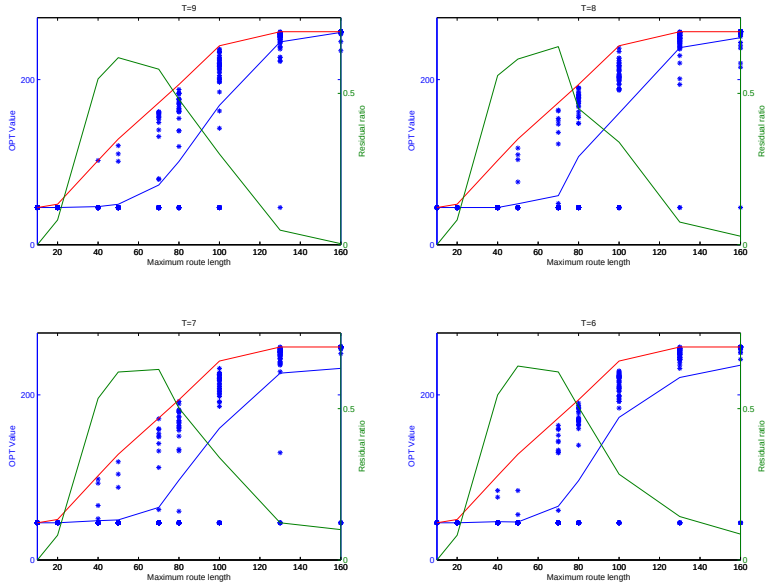



Figure D.20: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

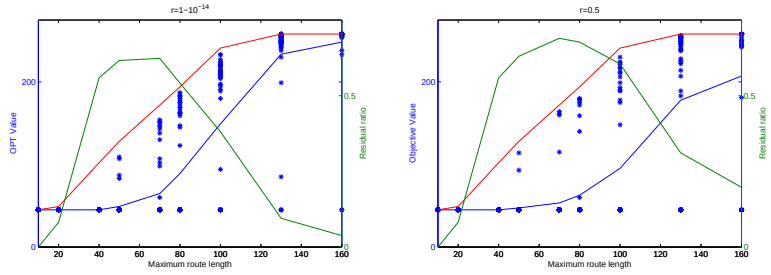



Figure D.21: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

### D.2.2 Results for reduction factor, $r$, first run

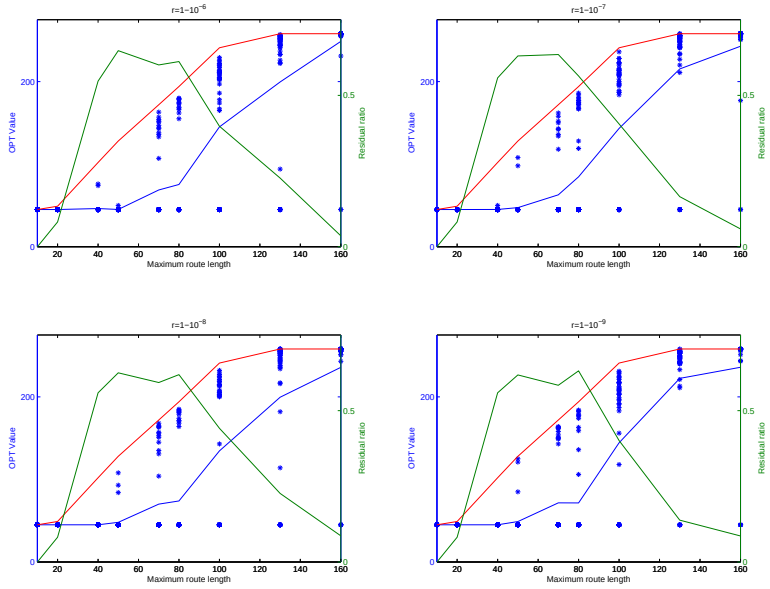



Figure D.22: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

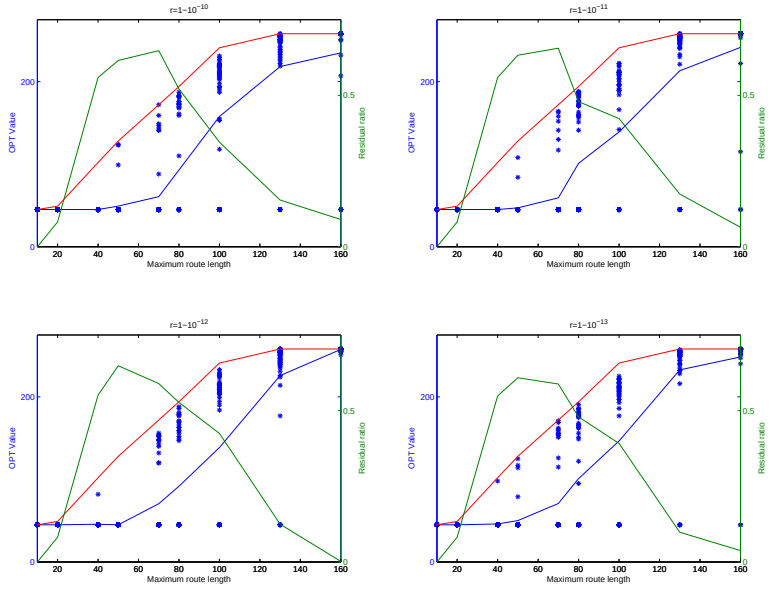



Figure D.23: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

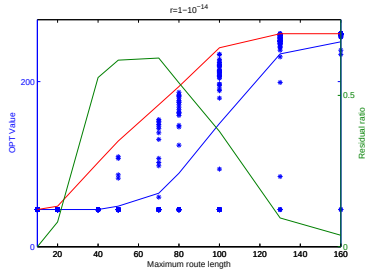



Figure D.24: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

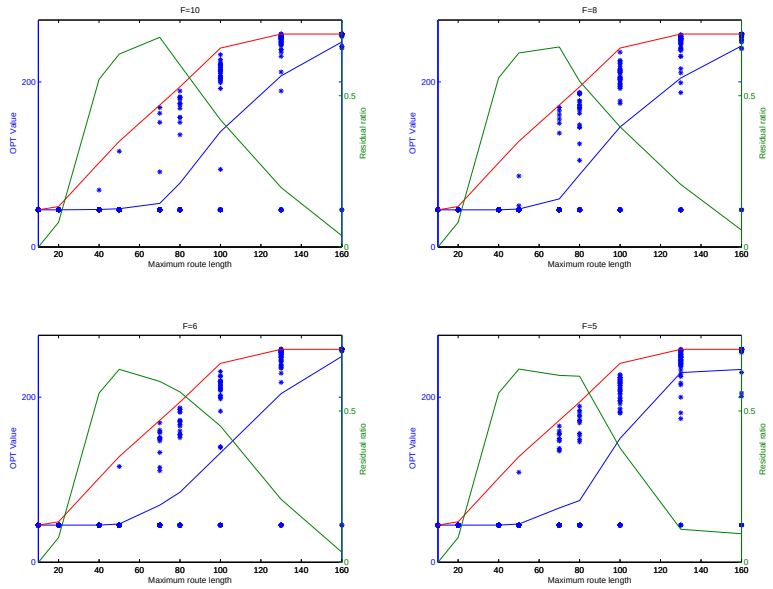



Figure D.25: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

### D.2.3 Results for stopping criteria, $F$, first run

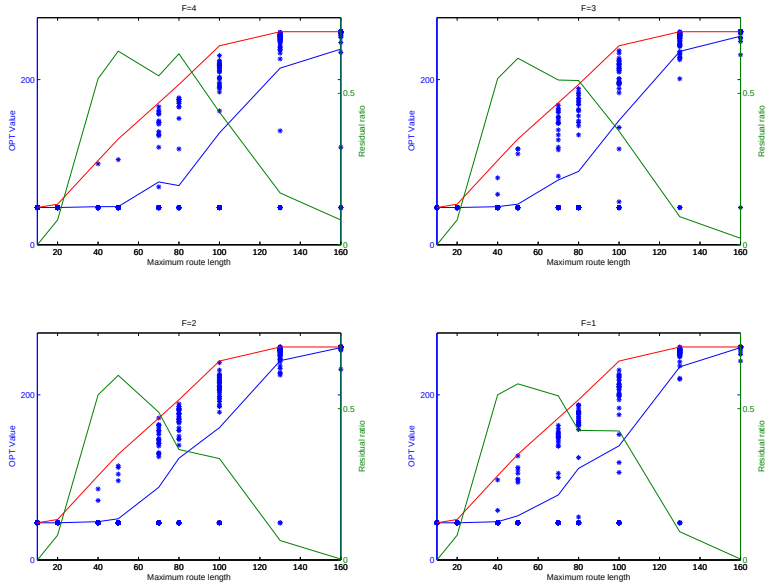



Figure D.26: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

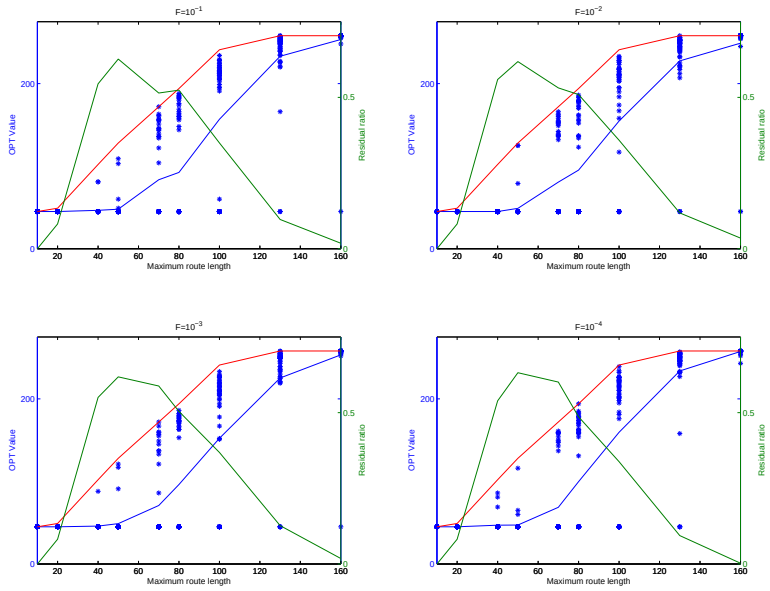



Figure D.27: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

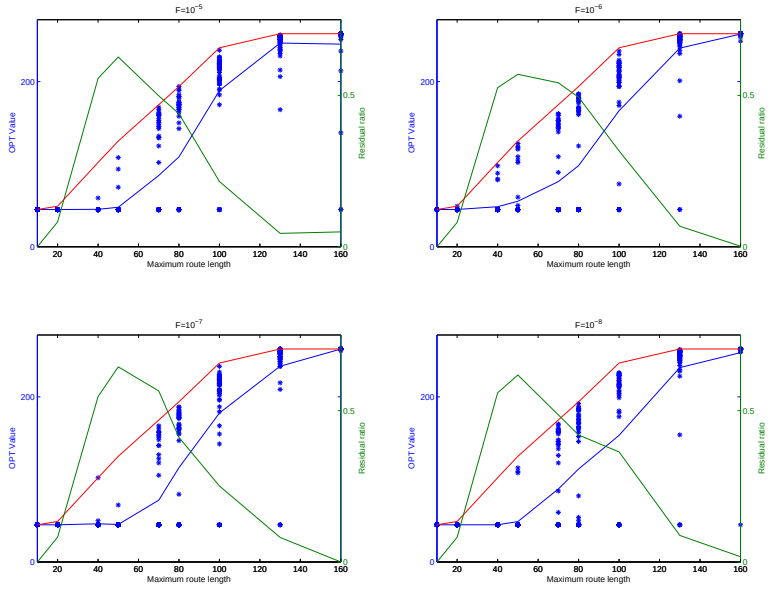



Figure D.28: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

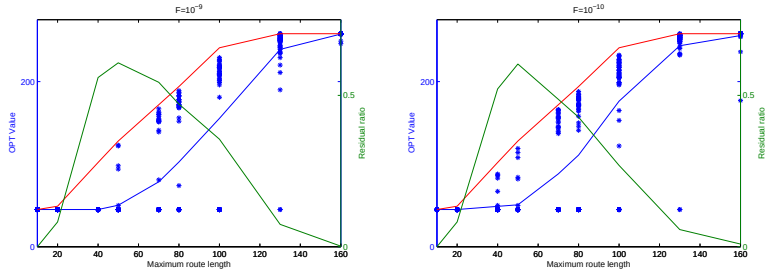



Figure D.29: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

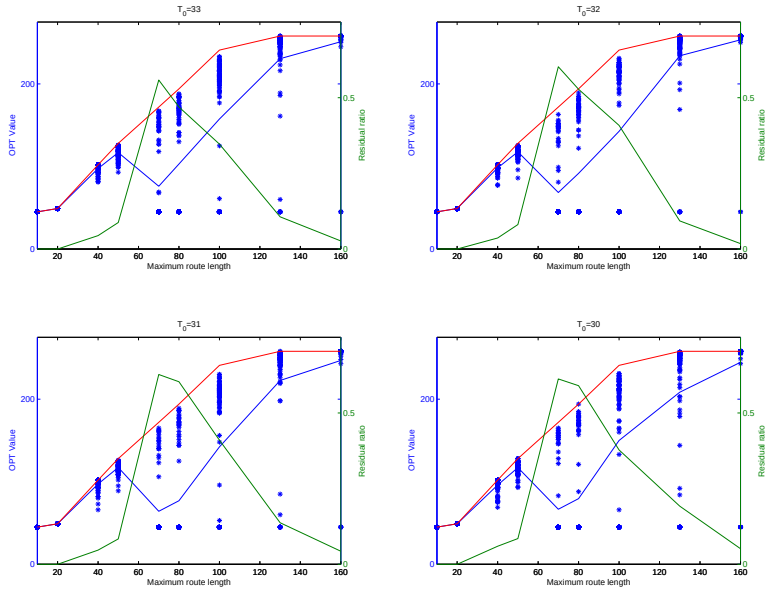



Figure D.30: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

### D.2.4 Results for temperature, $T$, second run

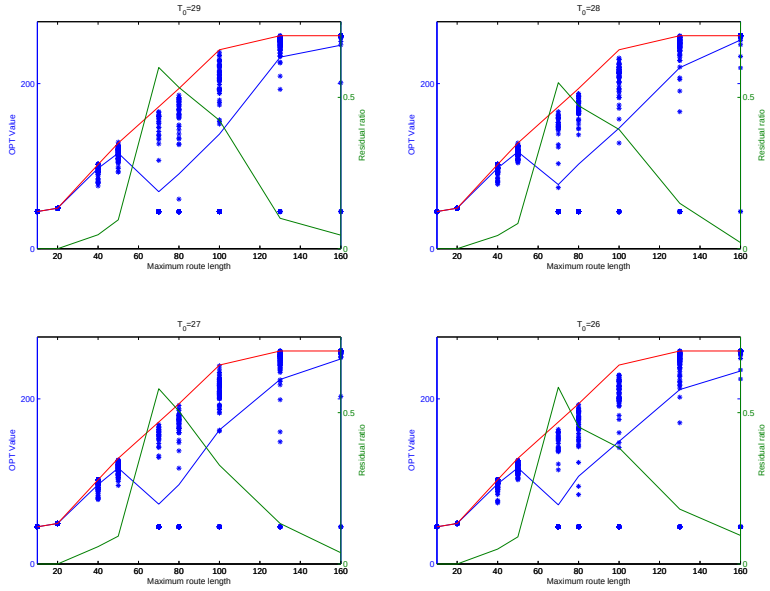



Figure D.31: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

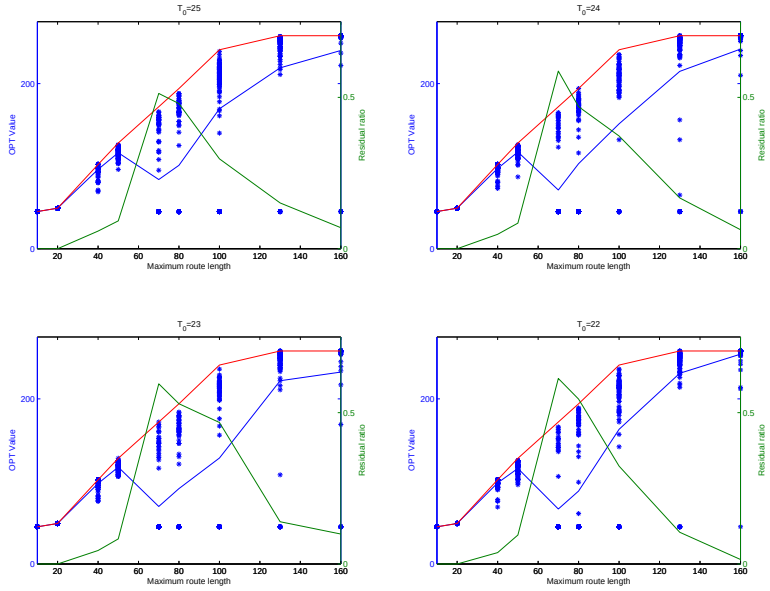



Figure D.32: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

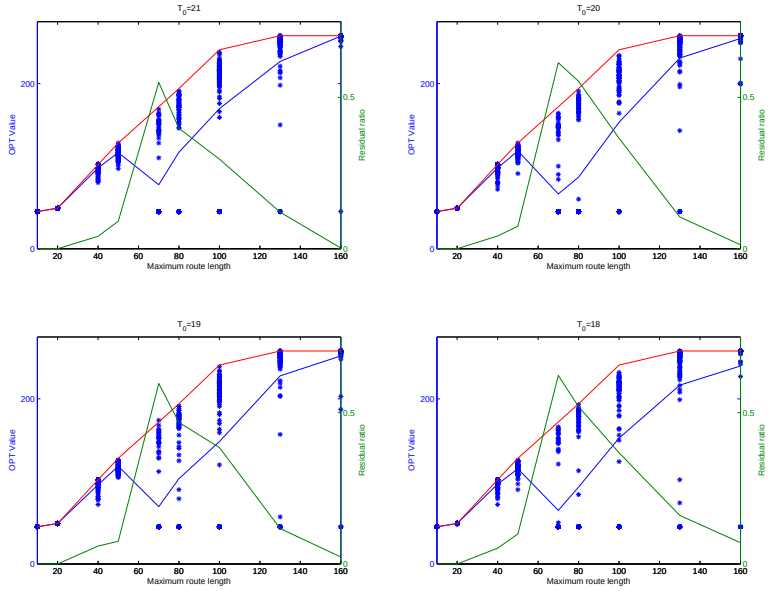



Figure D.33: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

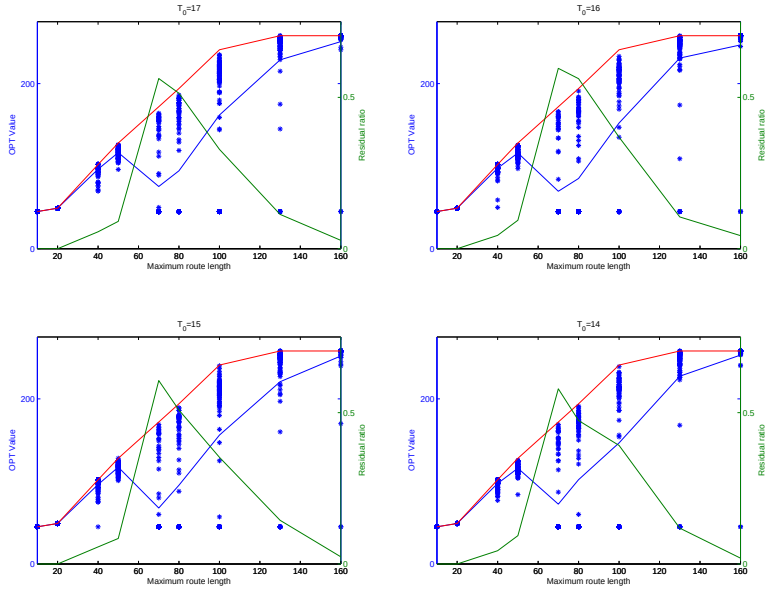



Figure D.34: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

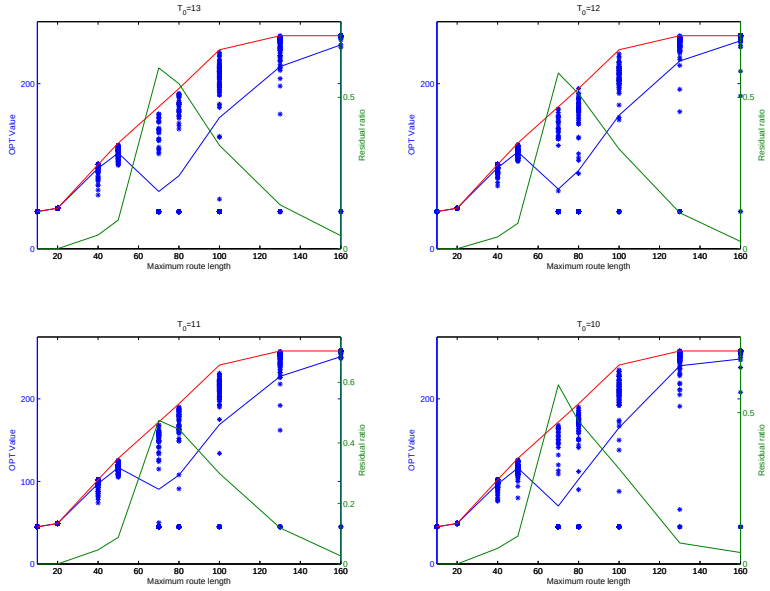



Figure D.35: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

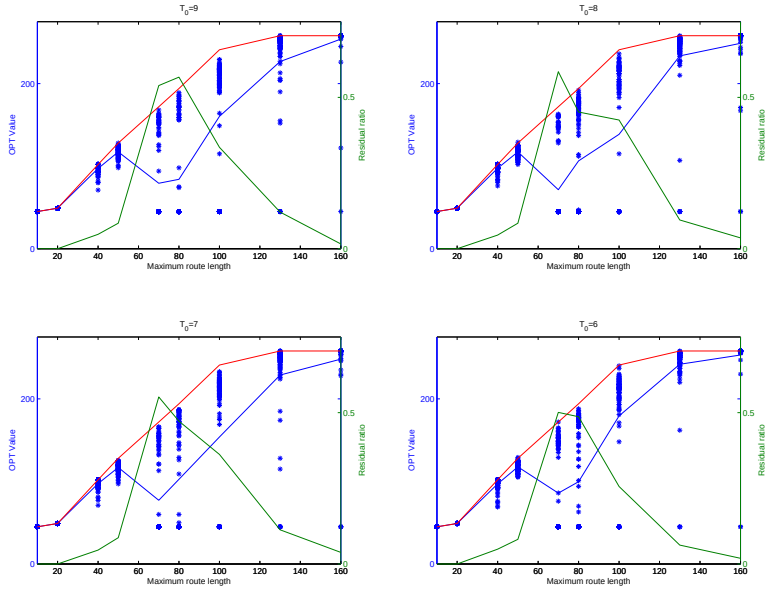



Figure D.36: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

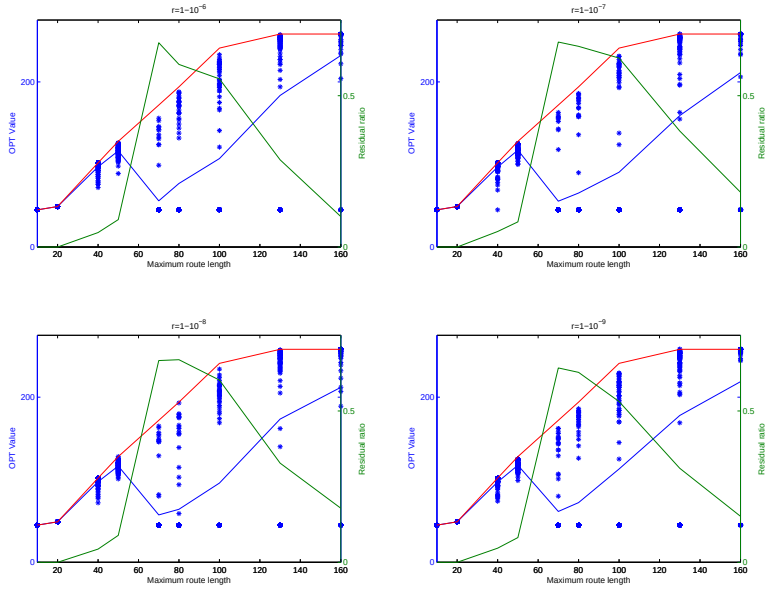



Figure D.37: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

### D.2.5 Results for reduction factor, $r$, second run

subsectionResults for stopping criteria, $F$, second run

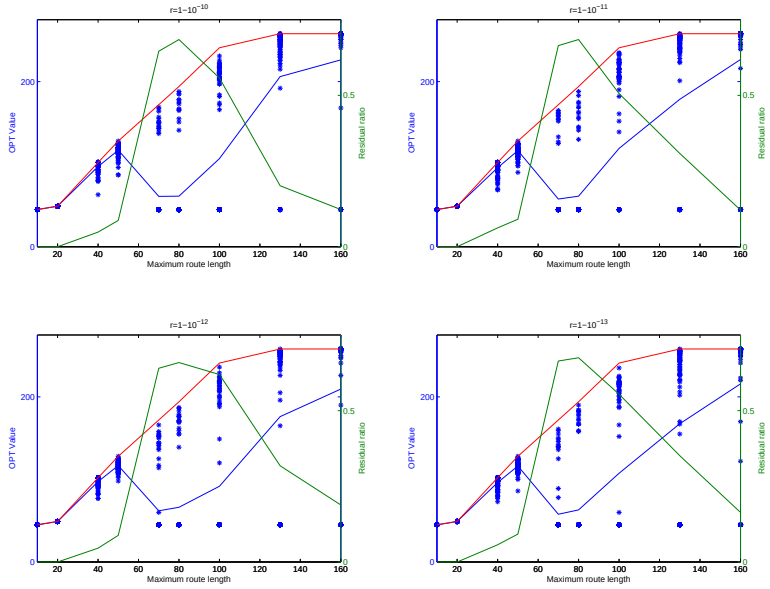



Figure D.38: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

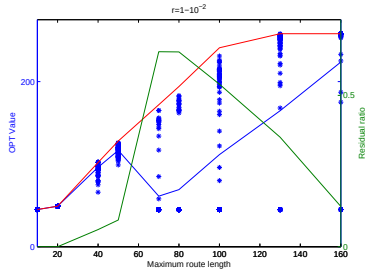



Figure D.39: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

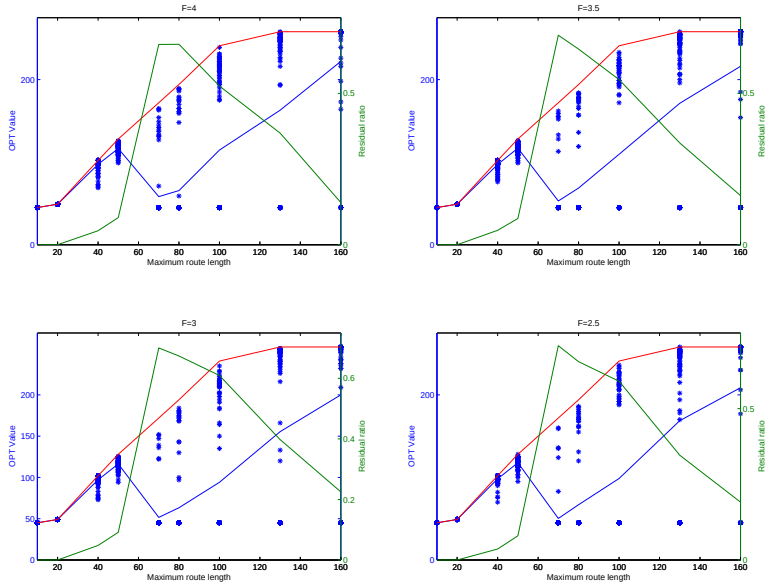



Figure D.40: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

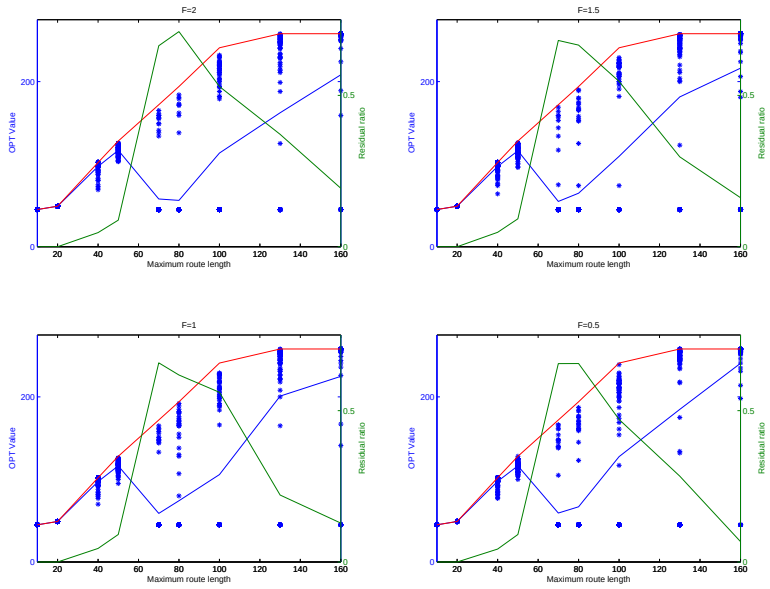



Figure D.41: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

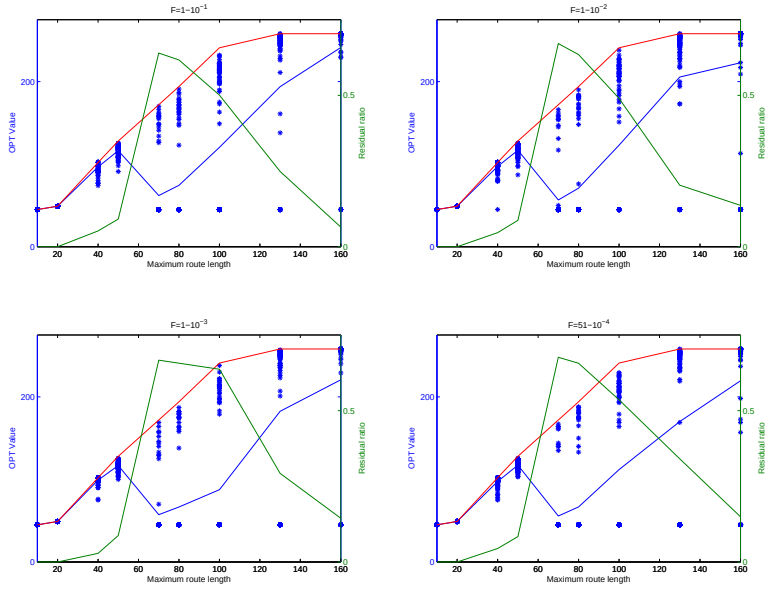



Figure D.42: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

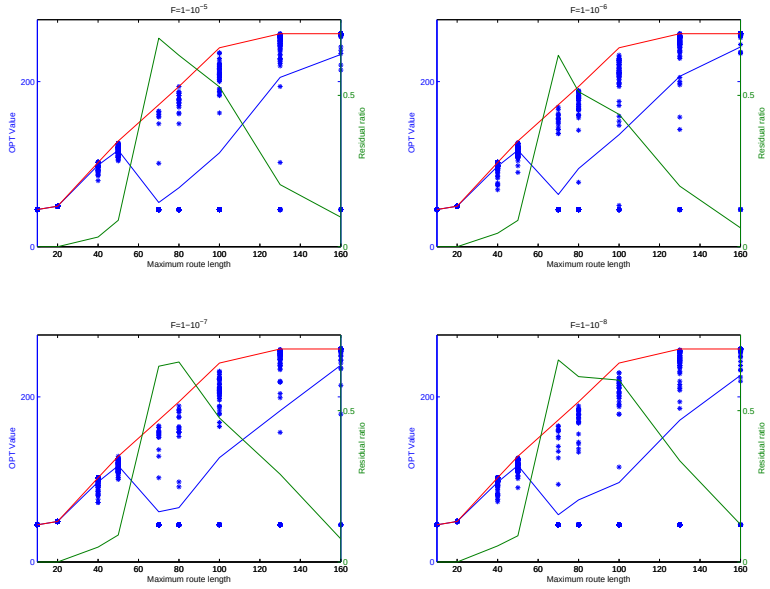



Figure D.43: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

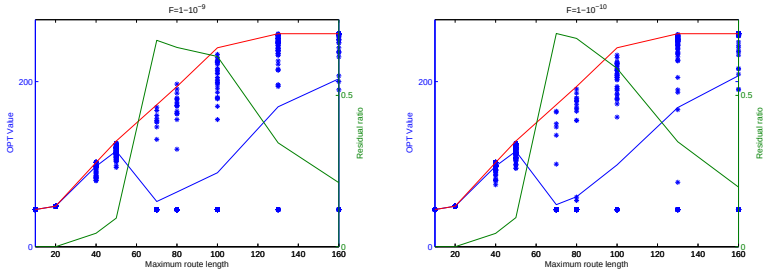



Figure D.44: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

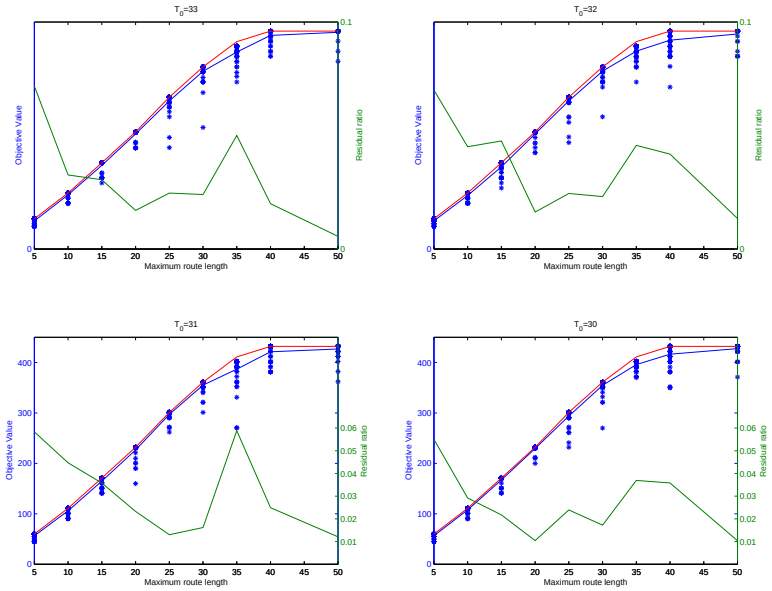



Figure D.45: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

### D.2.6 Results for Temperature, $T$, Data Set 3_50_$a$

### D.2.7 Results for Reduction Factor, $r$, Data Set 3_50_$a$

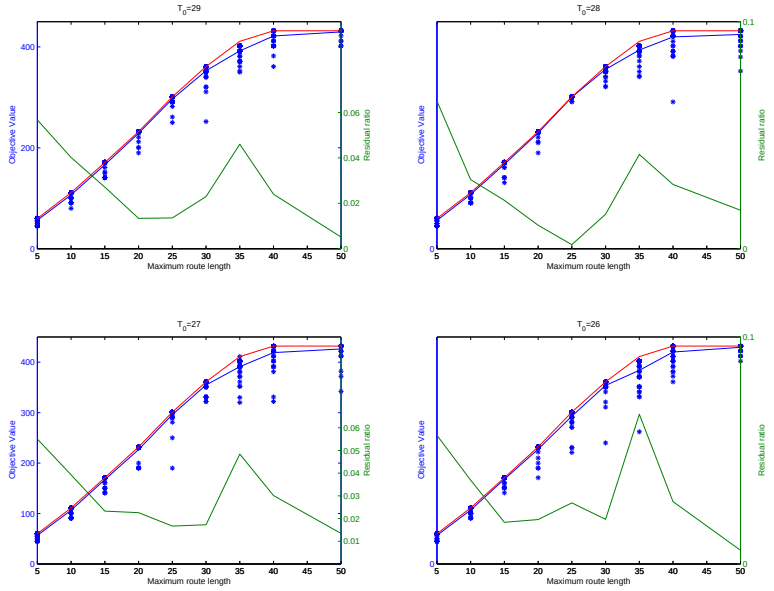



Figure D.46: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

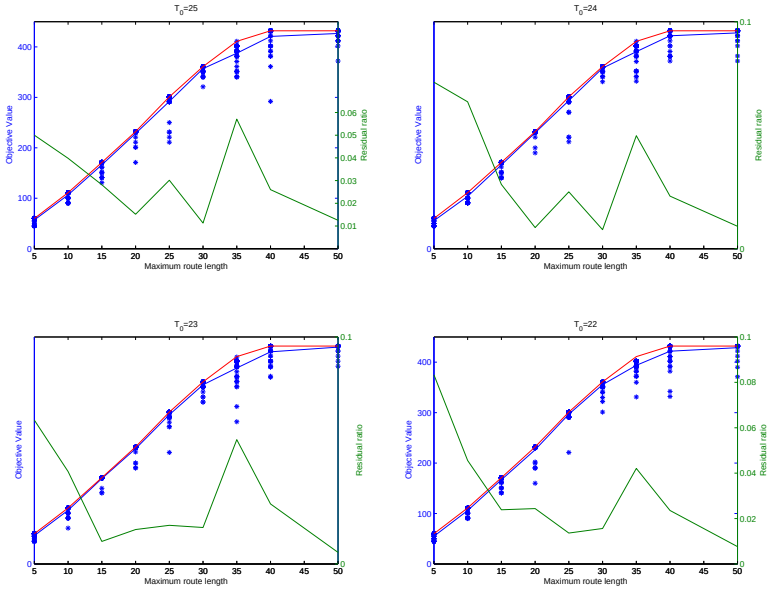



Figure D.47: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

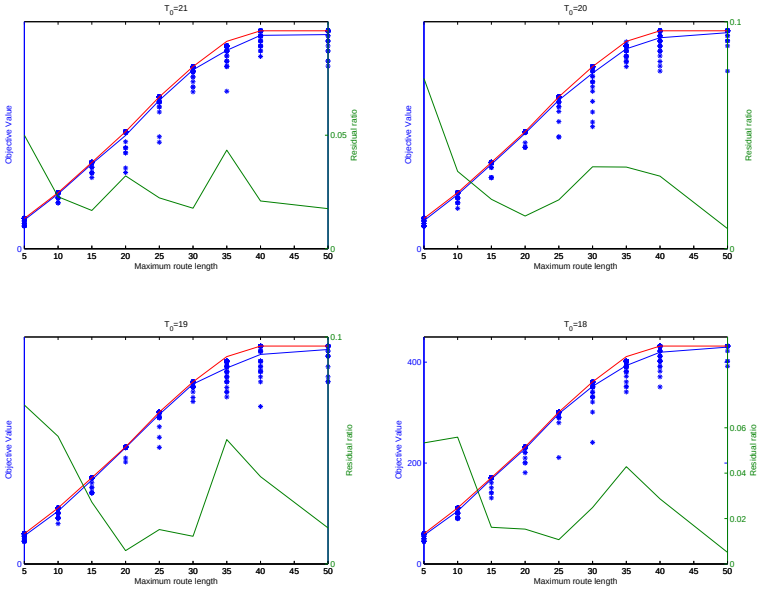



Figure D.48: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

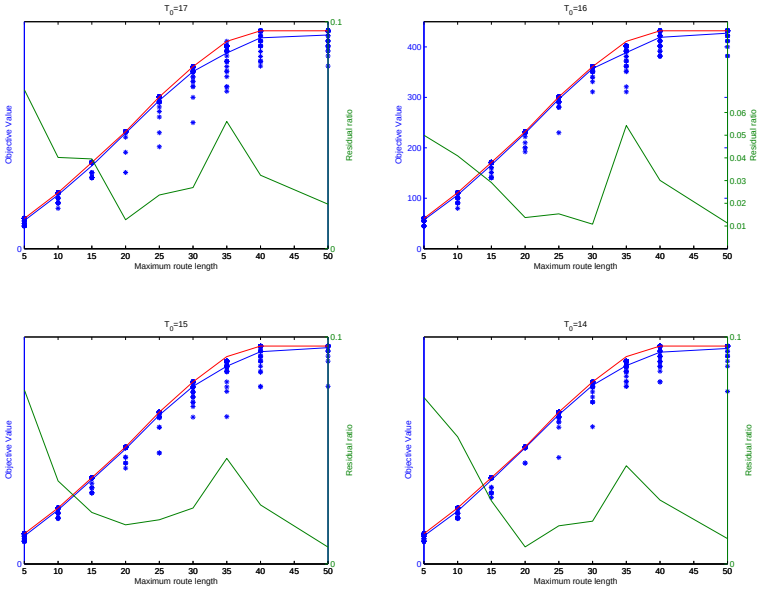



Figure D.49: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

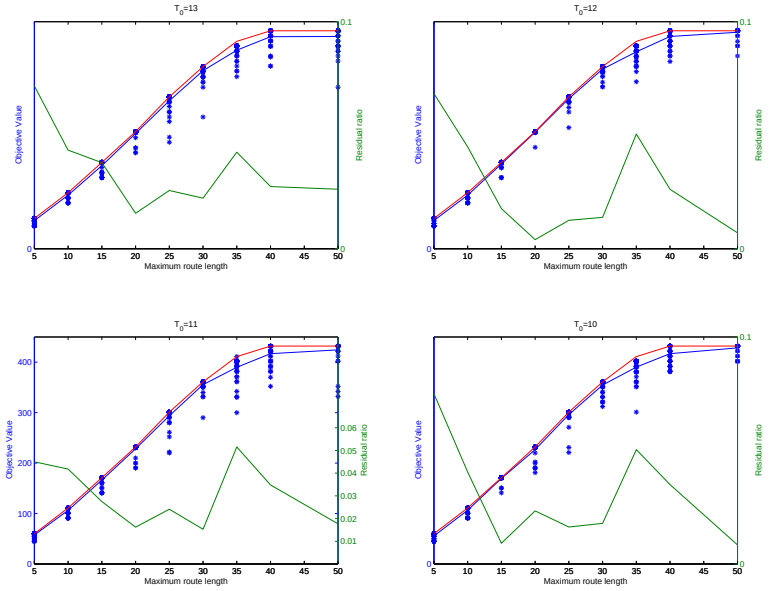



Figure D.50: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

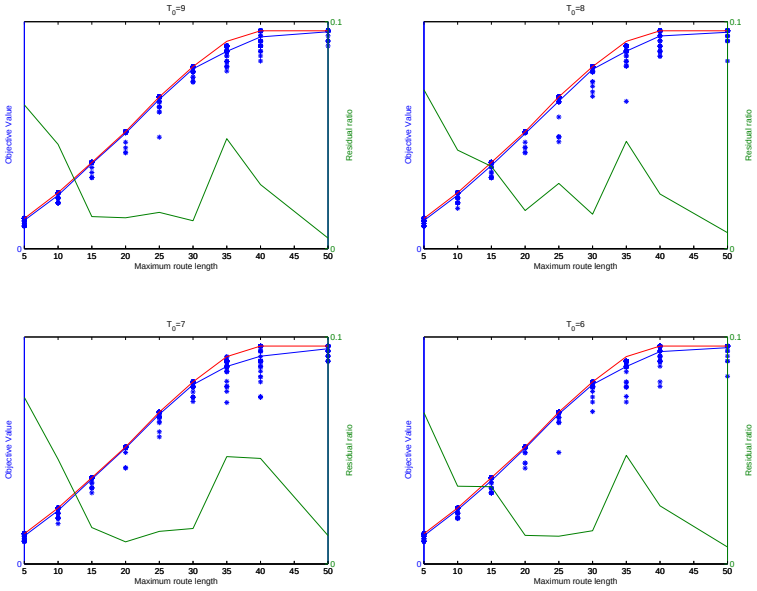



Figure D.51: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

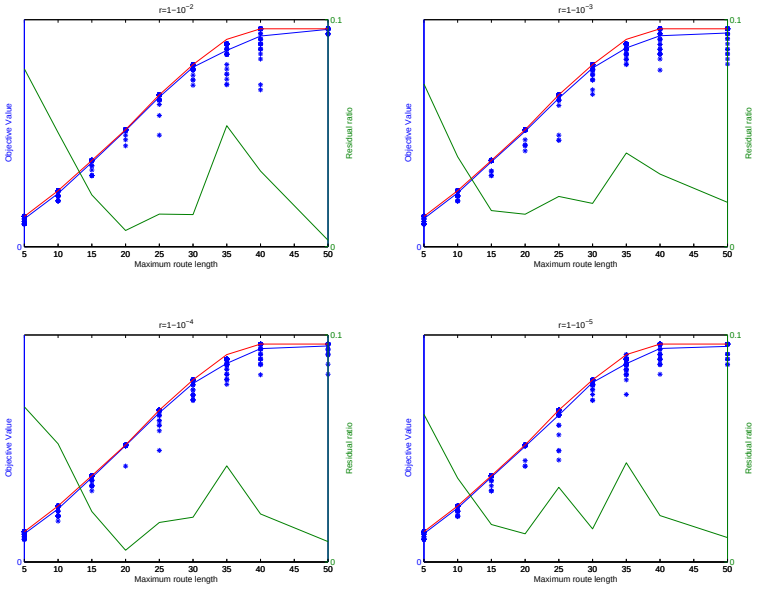



Figure D.52: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

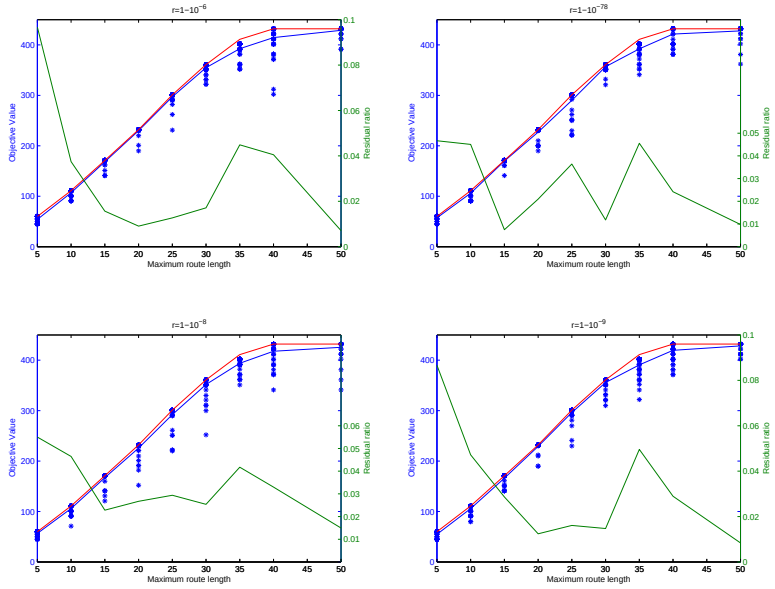



Figure D.53: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

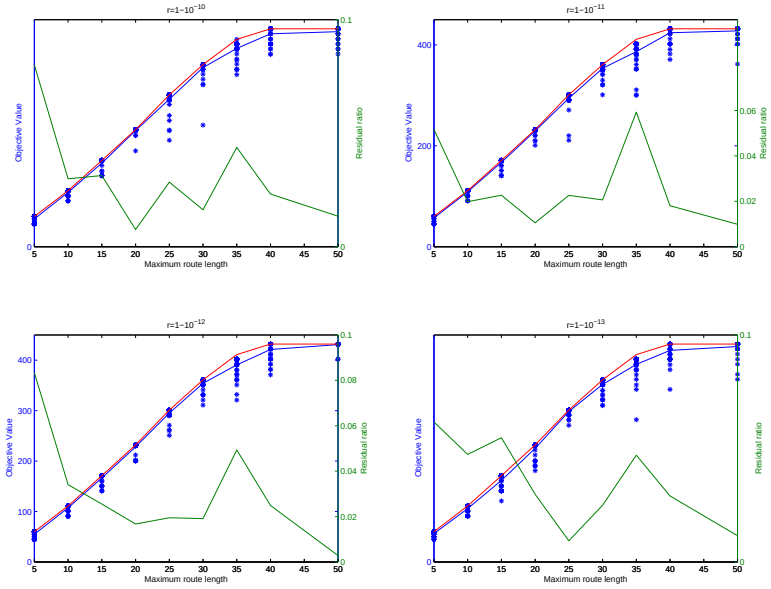



Figure D.54: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

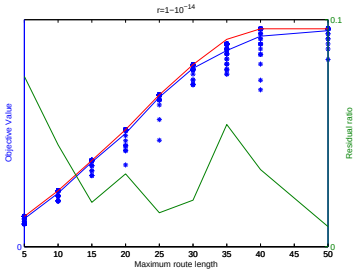



Figure D.55: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

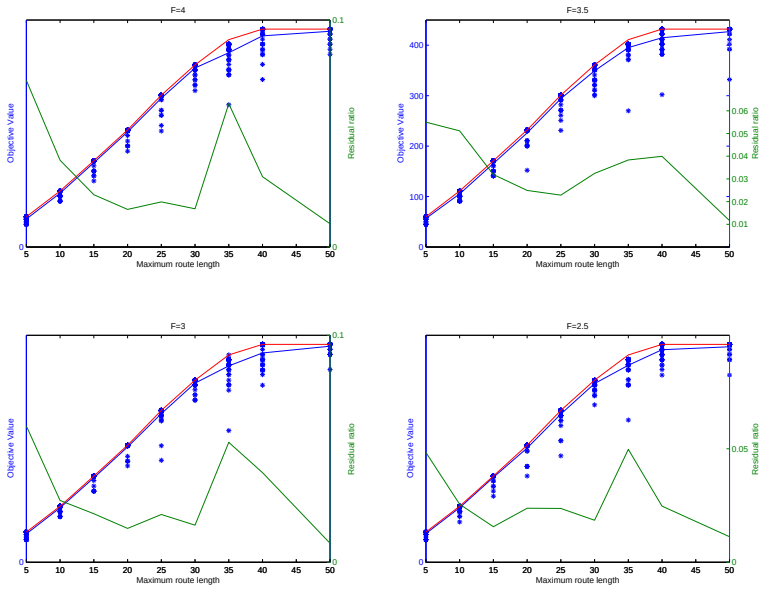



Figure D.56: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

### D.2.8 Results for Frozen Factor, $F$, Data Set 3_50_$a$

## D.3 Randomly Generated Data Sets

### D.3.1 50 point profit vector

0
3
9
2
4
9
3
3
5
5
7
6
5
5
10
10
6

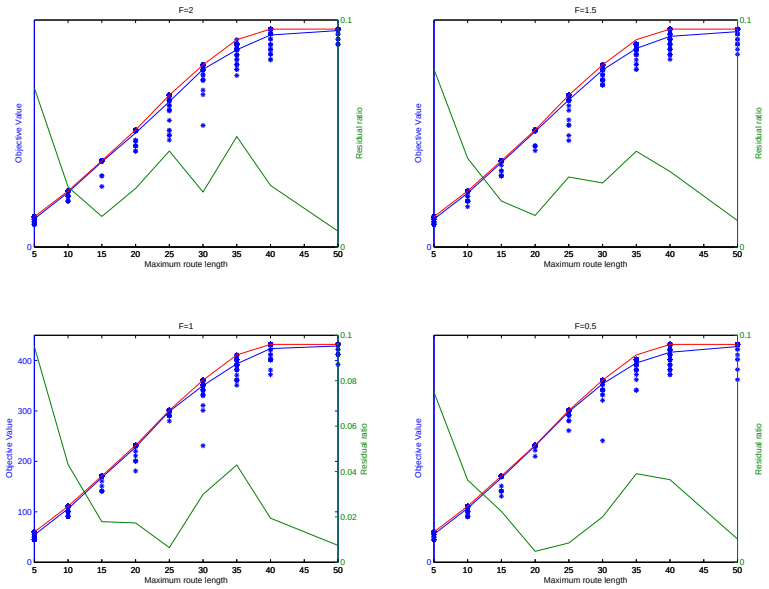



Figure D.57: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

8
4
5
1
10
7
2
3
10
8
5
6
9
5
3
5
2
3
8
4
1
2

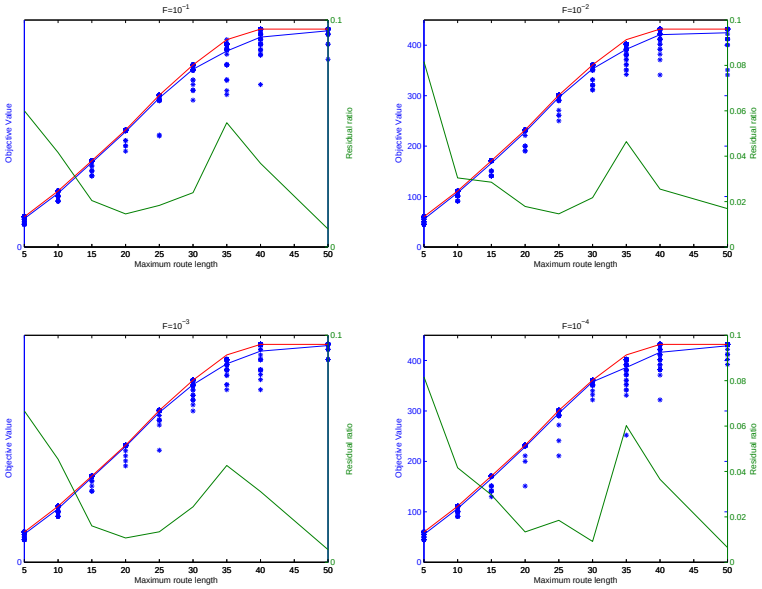



Figure D.58: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

1
4
2
10
2
7
10
2
8
1
1
7
0

### D.3.2 100 point profit vector

0
5
6
2
5
1

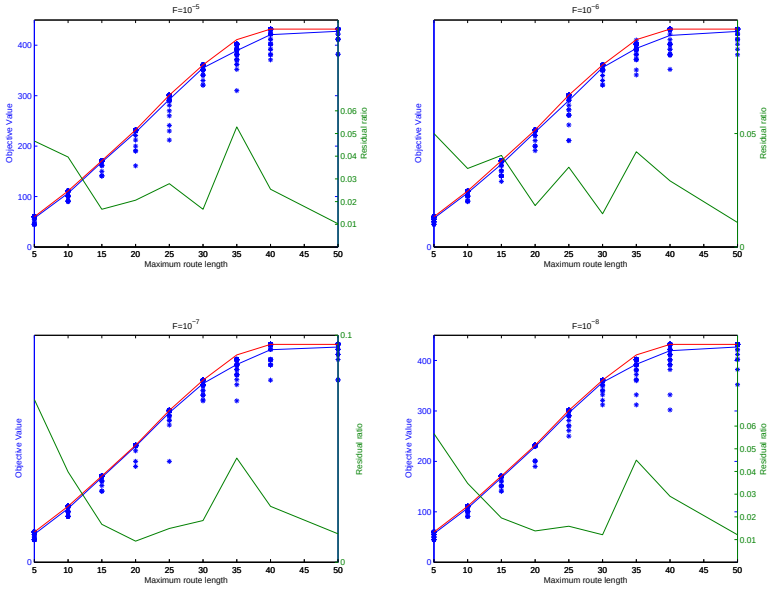



Figure D.59: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

10
9
6
8
9
5
5
10
2
10
1
6
5
3
1
1
3
6
9
8
2
3

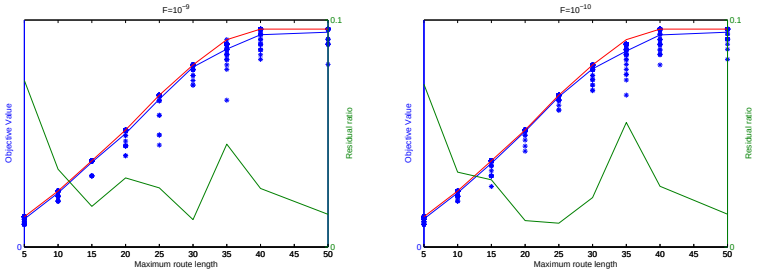



Figure D.60: This figues shows results for four different tempratues. Blues line and dots is the average value and the calculated optimas, the red lin is the best known optimum and the green line is the residual ratio.

2
7
10
8
1
1
6
5
3
5
8
2
5
6
1
9
5
2
4
6
10
1

9
6
6
7
1
6
2
3
9
2
1
10
1
9
6
3
8
7
10
8
7
8
2
1
1
2
10
2
10
1
9
5
1
10
8
5
8
4
8
5
2
2
5
6
7
7
10

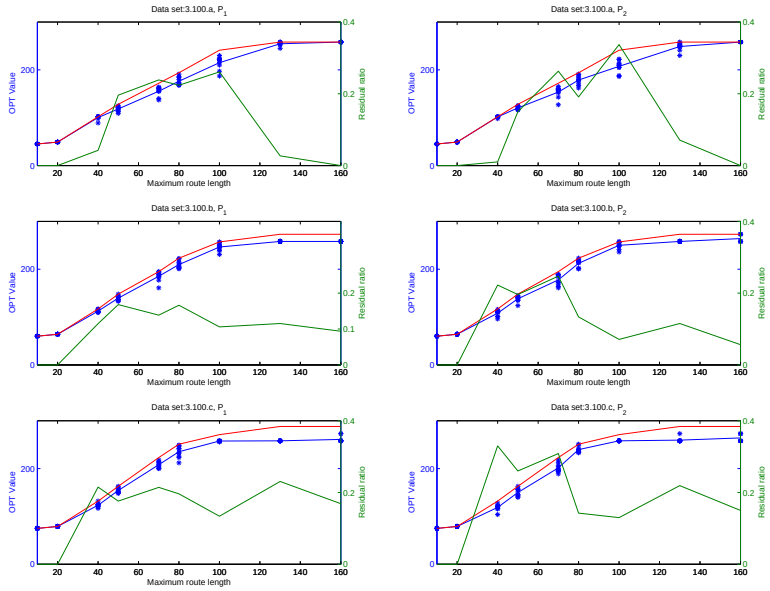
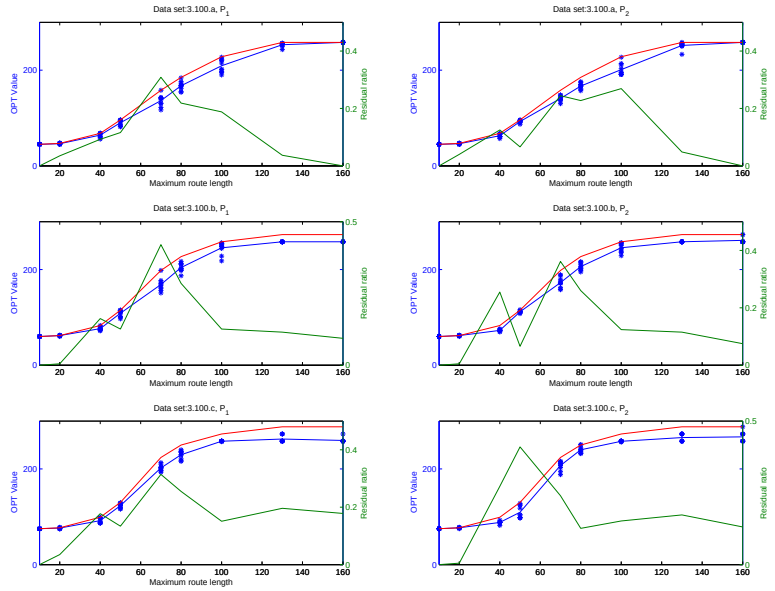



Figure D.61: Blues line and dots is the average values and the calculated optimums, the red line is the best known optimum and the green line is the residual ratio.

10
5
5
7
0

### D.3.3 Test with data sets 50a,b,c,d and e with 450,000 Iterations




,

Figure D.62: Blues line and dots is the average values and the calculated optimums, the red line is the best known optimum and the green line is the residual ratio.

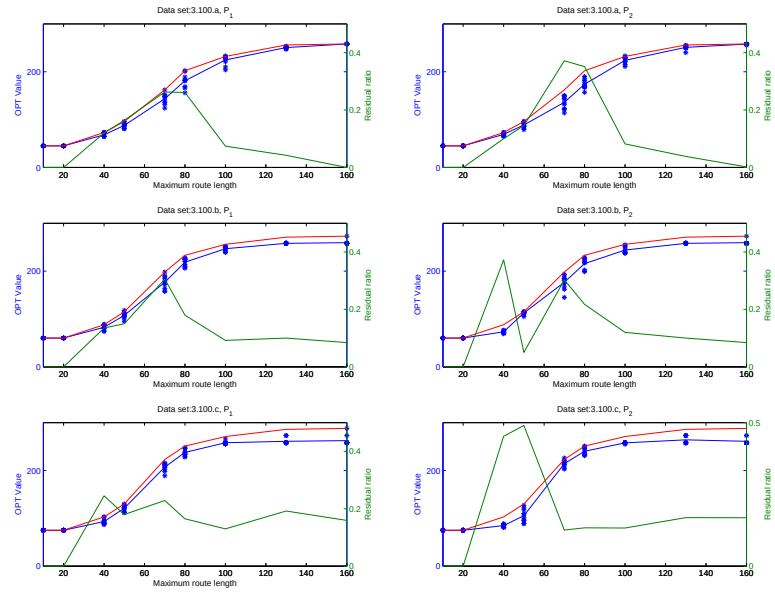



Figure D.63: Blues line and dots is the average values and the calculated optimums, the red line is the best known optimum and the green line is the residual ratio.

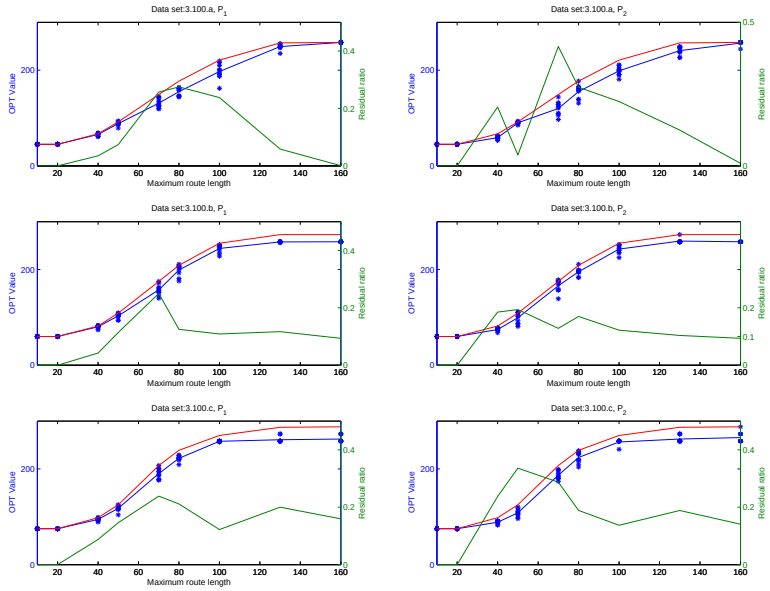



Figure D.64: Blues line and dots is the average values and the calculated optimums, the red line is the best known optimum and the green line is the residual ratio.

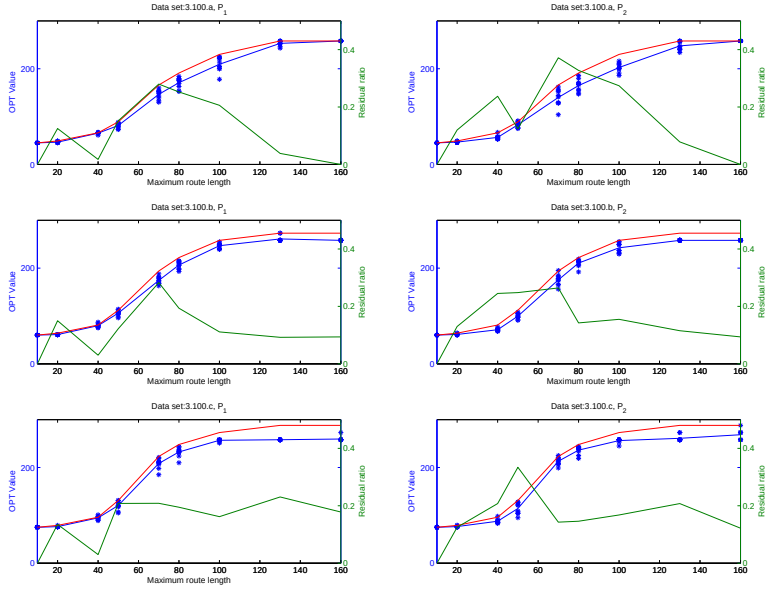



Figure D.65: Blues line and dots is the average values and the calculated optimums, the red line is the best known optimum and the green line is the residual ratio.

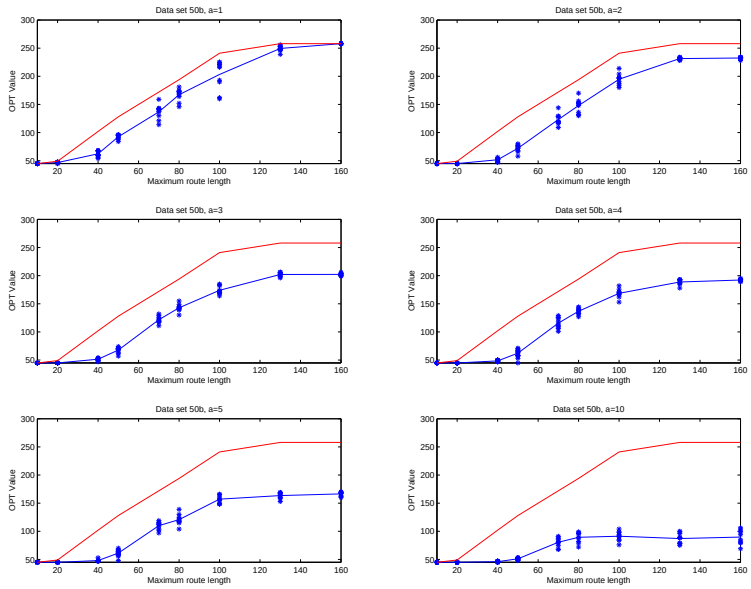



Figure D.66: Blues line and dots is the average values and the calculated optimums and the red line is the best known optimum

## D.4 Results Comparison to Decrease.java with New Cooling Schedule

| **M** | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | $\|K\|$ | $p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **OPT** | 45 | 49 | 49 | 59 | 69 | 76 | 95 | 107 | 110 | 3 | $p_2$ |
| **AVE** | 45 | 48.2 | 49 | 50.45 | 64.55 | 73.3 | 89.05 | 99.2 | 107.45 | 3 | $p_2$ |
| **OPT** | 60 | 64 | 64 | 74 | 84 | 92 | 109 | 119 | 125 | 4 | $p_2$ |
| **AVE** | 60 | 63.6 | 63.8 | 65.9 | 80.75 | 86.55 | 101.65 | 111.3 | 113.35 | 4 | $p_2$ |
| **OPT** | 75 | 79 | 79 | 89 | 99 | 106 | 124 | 129 | 138 | 5 | $p_2$ |
| **AVE** | 75 | 79 | 78.8 | 80.45 | 95.85 | 101.15 | 118.85 | 122.85 | 125.85 | 5 | $p_2$ |

### D.4.1 Results for the Distance Constraint

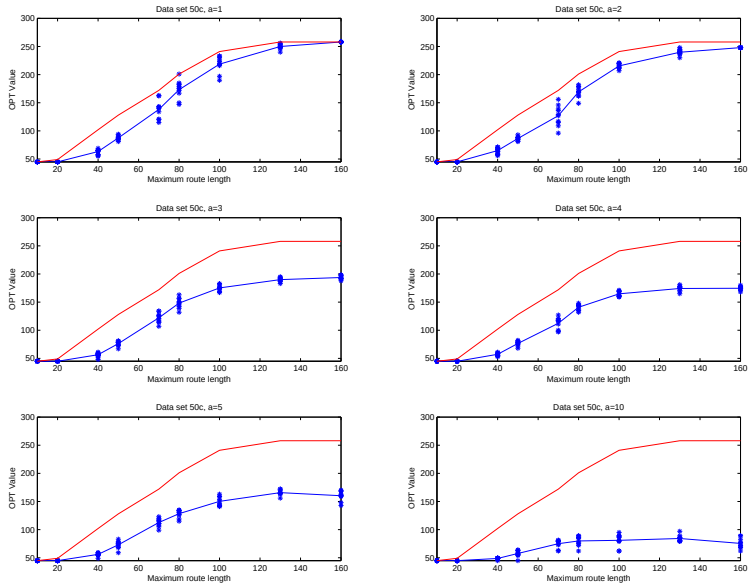



Figure D.67: Blues line and dots is the average values and the calculated optimums and the red line is the best known optimum

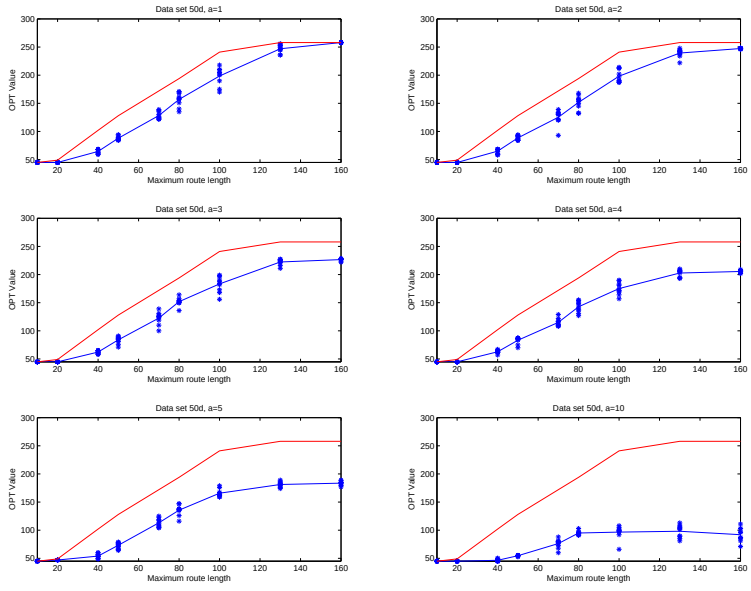



Figure D.68: Blues line and dots is the average values and the calculated optimums and the red line is the best known optimum

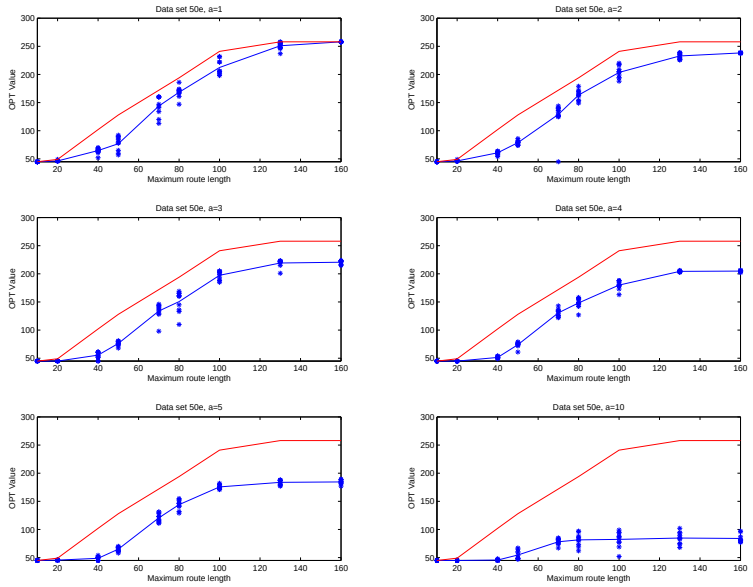



Figure D.69: Blues line and dots is the average values and the calculated optimums and the red line is the best known optimum

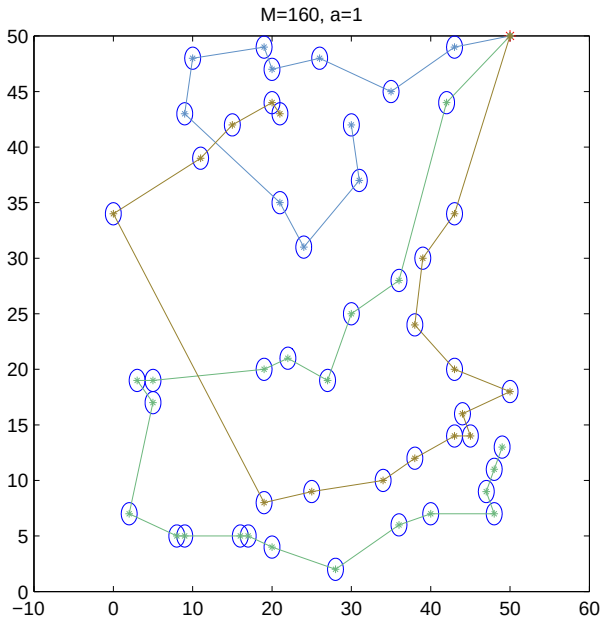



Figure D.70: Shows the routes constructed when $a = 1$ and $M = 160$. The circles are the area where that must be traveld before another pick up point is chosen.

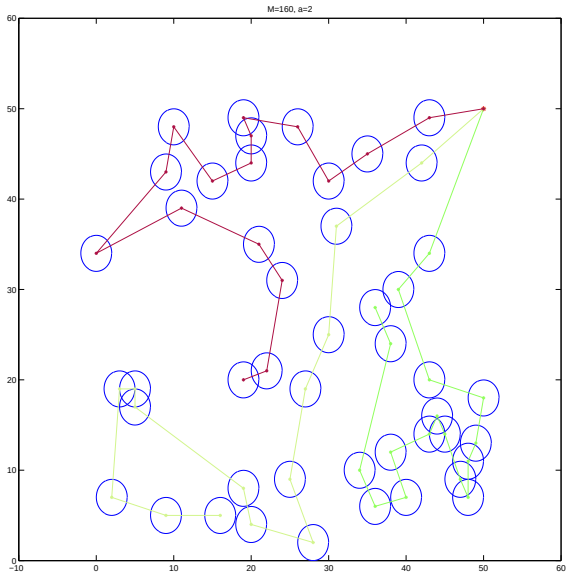



Figure D.71: Shows the routes constructed when $a = 2$ and $M = 160$. The circles are the area where that must be traveld before another pick up point is chosen.

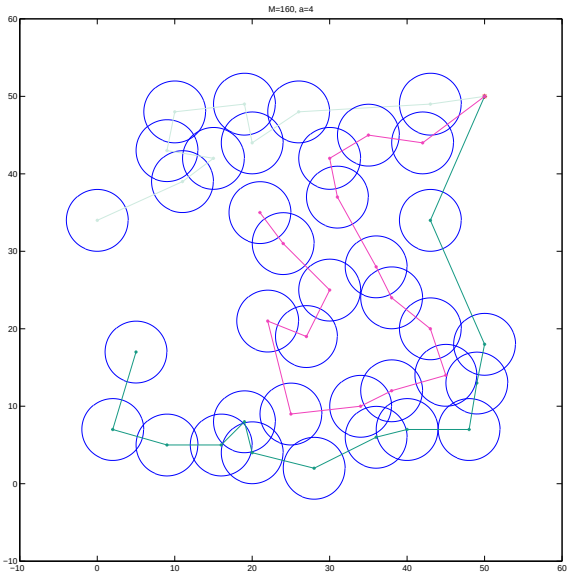



Figure D.72: Shows the routes constructed when $a = 4$ and $M = 160$. The circles are the area where that must be traveld before another pick up point is chosen.

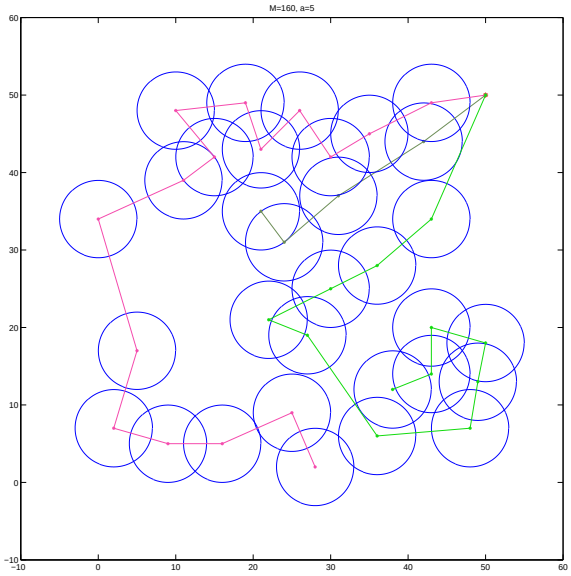



Figure D.73: Shows the routes constructed when $a = 5$ and $M = 160$. The circles are the area where that must be traveld before another pick up point is chosen.