Erasmus University Rotterdam
Erasmus School of Economics

Bachelor Thesis Econometrics and Operations Research

Operations Research and Quantitative Logistics

Meta-heuristic for the mobile facility location problem

Master process with look ahead mechanisms for efficient and high quality solutions

ARTICLE INFO

Author: Koen Vermeulen
Student number: 485054

Supervisor:
MSc. Lisanne van Rijn

Second assessor:
Dr. Twan A.B. Dollevoet

Keywords:
Mobile facility location
Integer programming
Local search heuristic
Meta-heuristic
Master process
Look ahead mechanisms

ABSTRACT

In the mobile facility location problem (MFLP), the relocation of a set of existing facilities and the assignment of clients to these facilities are considered. The optimal solution of the MFLP corresponds to the solution with the minimum sum of facility movements and client travel costs. Because this combinatorial optimization problem is concluded to be $\mathcal{NP}$-hard, computation times of MFLP integer programming (IP) formulations grow drastically by increasing problem instance sizes. Therefore, this thesis studies different formulations and heuristics, searching for a solution method that provides high quality MFLP solutions in relatively short computation times. Firstly, different existing IP formulations and local search heuristics of the MFLP are provided and performed. From the results, the hybrid local search is confirmed to be the best current heuristic of the MFLP, combining the speed and solution quality of two other local search heuristics. Secondly, a new hybrid meta-heuristic is developed, consisting of an iterative master process with intelligent look ahead mechanisms incorporated. For every possible step, this meta-heuristic solves the hybrid local search and chooses the step with the most promising final solution. Different variants of this new meta-heuristic are developed. Finally, by comparing the results of all solution methods and making the trade-off between solution quality and computation time, one variant of the new meta-heuristic is concluded to be the new best solution method for quickly computing high quality MFLP solutions.

July 5, 2020

Contents

1 Introduction

Worldwide, companies are focusing on development and growth in order to become more profitable. At the same time, companies are under the tremendous pressure of reducing the price of their products (Meyerson, 2001). Consequently, companies are searching for strategies to decrease the marginal costs of their products. In some cases, additional profits might be obtained by opening new facilities, such that the marginal costs could be decreased. For opening new facilities, one has to decide where to locate these facilities. This decision has to be made, considering different aspects depending on the problem, e.g. the location and capacity of the current facilities and the location and demand of the clients. The problem related to this decision is known as the *facility location problem* (FLP). The FLP considers the optimal location of facilities such that the total distance function between the facilities and the customers is minimized (Cornuéjols et al., 1983).

However, in many applications, cases arise in which facilities are already located but in an nonoptimal way, e.g. after takeovers of facilities or after client movements. In such cases, an optimal relocation has to be found for both the facilities and the clients. This relocation problem is referred to as the *mobile facility location problem* (MFLP). The MFLP considers the relocation of a set of existing facilities and the assignment of clients to these facilities such that the sum of facility movements costs and client travel costs is minimized.

Currently, there is a wide range of real life problems that are likely to be considered as a MFLP (Bespamyatnikh et al., 2000). Particularly, there are many applications with respect to mobile wireless communication networks. For example battlefield map servers and servers that provide support for network control could be considered mobile facilities. Therefore, insights into efficient solution methods for the MFLP are both of scientific relevance and of interest for practical applications (Bespamyatnikh et al., 2000). Currently, two *integer programming* (IP) formulations and three different heuristic approaches are described for the MFLP, introduced by Halper et al. (2015).

However, more research on heuristic solution methods may provide more effective solution methods for the MFLP. Therefore, the main focus of this thesis is to provide a solution method that computes high quality solutions for the MFLP relatively quickly. This leads to the following central research question:

Research question. *Which solution method provides high quality solutions for the MFLP in relatively short computational running times?*

The thesis is divided into a two-phase structure. In the first phase, the MFLP is studied by partly reproducing the research previously conducted by Halper et al. (2015). In this reproduction phase, two IP formulations for the MFLP are provided. Besides, the decomposition of the MFLP into two subproblems is discussed (Halper et al., 2015). Based on this result, two *local search heuristics* of Halper et al. (2015) are implemented. Local search heuristics explore a neighborhood of solutions and iteratively improve the solution.

The results of the first phase confirm the research of Halper et al. (2015) by concluding the hybrid local search to be the best current heuristic for providing high quality MFLP solutions within relatively short computation times. However, the challenge of finding efficient and robust algorithms for the MFLP remains.

Therefore, in the second phase a new *meta-heuristic* is developed for the MFLP. This meta-heuristic is a master process consisting of intelligent look ahead mechanisms. In this solution method, a solution is built up by iteratively solving the hybrid local search of the first phase for every possible step and choosing the step with the most promising final solution. By making choices based on look ahead results, the meta-heuristic avoids the pitfall of making shortsighted steps based on immediate gains that could possibly lead to suboptimal choices. In this way, the meta-heuristic aims to provide solutions of high quality, improving the solutions of the local search heuristics.

Because of the wide range of different implementation possibilities of this meta-heuristic, six different variants are implemented. From all implemented solutions methods, four candidates are proposed as potential best solution method for the MFLP. For every candidate, the trade-off between solution quality and computation time had to be considered. From these trade-offs, one variant of the meta-heuristic is indicated to provide solutions with the best balance between solution quality and time efficiency. This meta-heuristic variant has incorporated an iteration time restriction of 0.001 seconds and iterates in a fixed order. The solution method has provided reproducible solutions of high quality with a maximum deviation of 2% from the optimal solution, computed in relatively short computation times of on average 7 minutes. By comparing all results, this meta-heuristic variant is concluded to be the best new solution method for providing companies and organizations high quality MFLP solutions within short computation times.

In the remainder of this paper, the following structure is followed. In Section 2 the problem description of this thesis is described. A review of relevant literature is provided in Section 3. The methodology of the thesis is discussed in Section 4. Here, the integer programming (IP) problems, local search heuristics and the new developed meta-heuristic are described and formulated. Section 5 presents information about the database and the different data instances. In Section 6 the results of multiple problem instances are provided for the IP formulations and for different versions of the local search heuristics and meta-heuristic. Finally, the conclusion and further research of this thesis are discussed in Section 7 .

2 Problem description

In this research project, the mobile facility location problem (MFLP) is studied, originally introduced by Demaine et al. (2009). The MFLP is proposed by transforming the formulation of the facility location problem (FLP). The FLP is a classical operations research problem and is defined here (Cornuéjols et al., 1983).

Definition 1. *The facility location problem (FLP) considers the location of a set facilities such that such that the total distance function between the facilities and the customers is minimized.*

Based on this classical problem, the MFLP relocates a set of existing facilities with an initial location, instead of locating them initially. Besides, the MFLP assigns the set of clients to the new facility destinations. A definition for the MFLP is required and provided here.

Definition 2. *The mobile facility location problem (MFLP) considers the relocation of a set of existing facilities and the assignment of clients to these facilities such that the total sum of facility movement costs and client travel costs is minimized.*

The MFLP is formulated on a graph $G(V, E)$, where the set V is denoted as the set of vertices and set E is denoted as the set of edges, containing edges with all positive costs. Furthermore, there are facilities and clients initially positioned on the graph. The facilities are positioned at subset $F \subseteq V$ and the clients are positioned at the subset $C \subseteq V$. The objective function that is minimized in the MFLP equals the total sum of the weighted distances traveled by the facilities and clients together.

Figure 1 presents a facility and client assignment in a framework consisting of three layers. From top to bottom, the first layer presents the initial facility locations, the second layer presents the new and possible facility destinations[1] and the third layer presents the clients. Finally, the weight on every edge represent the distance between the two vertices connected by this edge.

[1]The possible facility destinations are referred to as vertices.

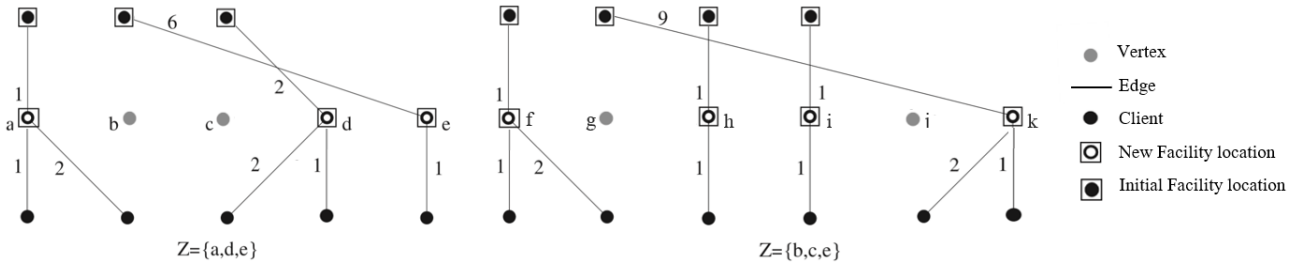

Figure 1: Example of an allocation of facilities and clients.

3 Literature

In this section, relevant literature regarding the MFLP is discussed. In Section 3.1 two IP formulations of the MFLP are discussed. Then, three local search heuristics are described in Section 3.2. Afterwards, in Section 3.3 the *pilot method* is introduced, looking ahead by calculating *pilot* solutions for every swap, performing the swap of the best pilot solution. Finally, literature of the capacitated MFLP is described in Section 3.4.

3.1 IP formulations

For solving the MFLP, different integer problems are described. Friggstad and Salavatipour (2011) provide an exact IP formulation for the MFLP and conclude that each facility has a unique destination. Namely, if multiple facilities share the same destination vertex, all but one of the facilities could instead be kept at their initial locations, without increasing the objective function. Halper et al. (2015) use this result to decompose the MFLP into two subproblems: the *facility assignment (FA)* and *client assignment (CA)* subproblem. Halper et al. (2015) use this decomposition observation to provide a new IP formulation for the MFLP, using less memory such that larger-scaled instances can be solved optimally in relatively shorter computation times.

3.2 Local search heuristics

Generally, combinatorial optimization problems turn out to be $\mathcal{NP}$-hard (Garey and Johnson, 1979). Therefore, it can concluded to be challenging to find efficient algorithms for the combinatorial optimization problem MFLP (Voßs et al., 2005). Nevertheless, three different local search heuristics for the MFLP are described in the research of Halper et al. (2015).

The first local search heuristic is based on a collection of local search operations defining a solution neighborhood (Friggstad and Salavatipour, 2011). Within this neighborhood, all possible swaps between the initial facility locations and vertices are investigated and compared. Finally, the best swap found is performed.

The second local search heuristic is based on a decomposition of the MFLP into the facility assignment (FA) and client assignment (CA) subproblem (Halper et al., 2015). This local search explores the neighborhood of solutions by solving the FA and CA subproblems optimally for every possible swap and performing the best swap after each iteration.

Finally, the third local search heuristic provide solutions by combining the speed of the first local search with the capability of the second local search to resolve the facility assignment subproblem optimally. Following the solving procedure of the second local search, the third local search solves a least cost perfect matching for every possible swap instead of the complete facility assignment. Finally, this local search resolves the complete facility and client assignment at the end of the procedure to check for further possible improvements.

3.3 Pilot method

Voßs et al. (2005) describe *meta-heuristics*, combining local search concepts with superordinate mechanisms. Meta-heuristics are iterative master processes guiding and modifying operations and iterations of a subordinate heuristic, producing efficient and high-quality solutions (Voßs et al., 2005).

Generally, algorithms are performed by making greedy choices based on a specific heuristic measurement (Rayward-Smith and Clare, 1986). However, most heuristic measurements are determined shortsighted by means of an immediate gain, which possibly lead to suboptimal choices (Voßs et al., 2005). Therefore, researchers have incorporated look ahead features in heuristic measurements, especially in artificial intelligence communities (Pearl, 1984). Here, intelligent mechanisms within heuristics measurements evaluate certain decisions based on the most promising solution in the nearby future, i.e. after a couple of iterations.

In this thesis, a new meta-heuristic is developed based on the concepts of the *pilot method* of Duin and Voß (1999). The ***P****reffered* ***I****terative* ***LO****ok ahead* ***T****echnique (pilot) method* defines a meta-heuristic computing a *pilot solution* for every choice (move), recording the best result. After computing all pilot solutions in the iteration, the *master solution* is extended by the best pilot solution. In this way, one cautiously builds up the master solution based on look ahead results (Voßs et al., 2005). Because the iterative choices in this meta-heuristic are based on look ahead mechanisms, this meta-heuristic can be referred to as a *tempered greedy algorithm*, *tempering* the greedy subordinate heuristic used.

Previous research of Voßs et al. (2005) has concluded that the pilot method is an effective meta-heuristic for avoiding greedy choices. However, the usage of look ahead mechanisms requires a considerable computation time. Therefore, Voßs et al. (2005) have recommended further research to investigate the best *evaluation depth* for the pilot method, where the evaluation depth is referred to as the parameter controlling the number of times the meta-heuristic restarts the heuristic procedure. Nevertheless, the extensive mechanisms of the pilot method are effective and could be valuable in the application of this thesis.

3.4 Capacitated MFLP

Finally, research is conducted on problem formulations extending the MFLP. One example is the mathematical formulation of the *Capacitated Mobile Facility Location Problem* (*CMFLP*) (Raghavan et al., 2019). The CMFLP can be defined as a MFLP in which facilities have a restricted capacity, such that the total demand of clients served by a specific facility has to be less or equal to the capacity of this facility. By considering the MFLP as a special case of the CMFLP in which all facilities have infinitely high capacities, the CMFLP can concluded to be a generalization of the MFLP. Consequently, the CMFLP is applicable to a very wide range of real life problems, making IP formulations and heuristics for this problem valuable.

Raghavan et al. (2019) provides two IP formulations for the CMFLP. The first formulation is based on a layered graph and extends the formulation introduced by Halper et al. (2015). The second formulation is formulated in a set partitioning framework. Furthermore, Raghavan et al. (2019) introduces two different heuristic approaches for solving the CMFLP. The first heuristic is a LP rounding heuristic. The second heuristic is an adaption of the second local search heuristic of Halper et al. (2015). However, despite the interesting applications of the CMFLP, it is considered beyond the scope of this thesis.

4 Methodology

In this section, different mathematical formulations, local search heuristics and a meta-heuristic for the MFLP are discussed. First, two exact mathematical formulations for the MFLP are provided in Section 4.1. Afterwards, the LP relaxation of this formulation is presented in Section 4.2. Then, a decomposition and a new framework for the MFLP are described in Section 4.3. Based on this description, in Section 4.4 different local search heuristics are explained. Finally, a new meta-heuristic for the MFLP is provided in Section 4.5.

4.1 Mathematical model

In this subsection, two mathematical formulations for the MFLP are described. Firstly, the sets, parameters and decision variables of the models are defined. Secondly, the mathematical models are formulated. Finally, the model descriptions, assumptions and a comparison between both models are provided.

Sets and parameters

The MFLP is formulated on a graph $G(V, E)$, where V is denoted as the set of vertices and E is denoted as the set of edges. Furthermore, facilities $F \subseteq V$ and clients $C \subseteq V$ are positioned initially on the graph. In Table 1, the different parameters for the IP formulations are described. Notice, the cost of moving a client or facility is proportional to the distance of this movement.

Table 1: Parameters in mathematical models

Parameter	Description
$d_{vv'}$	the distance of the shortest path from vertex $v \in V$ to vertex $v' \in V$ in the graph;
u_i	the per unit distance cost of satisfying the demand of client $i \in C$;
w_j	the per unit distance cost of relocating mobile facility $j \in F$;

Decision variables

The IP formulations make use of the decision variables presented in Table 2. Notice, the first two described decision variables are used for both formulations, while the third decision variable is only used in the second mathematical formulation.

Table 2: Decision variables in mathematical models

Variable	Description
x_{iv}	$\begin{cases} 1 & \text{if the destination of client } i \in C \text{ is vertex } v \in V, \\ 0 & \text{otherwise;} \end{cases}$
y_{jv}	$\begin{cases} 1 & \text{if the destination of facility } j \in F \text{ is vertex } v \in V, \\ 0 & \text{otherwise;} \end{cases}$
z_v	$\begin{cases} 1 & \text{if vertex } v \in V \text{ is the destination of some facility,} \\ 0 & \text{otherwise;} \end{cases}$

Mathematical formulation

The two integer programming formulations IP1 and IP2 are presented for the MFLP. The IP1 formulation is represented by (1)-(5) and the IP2 formulation is represented by (6)-(11). Notice, the objectives (1) and (6) as well as constraints (2)-(3) and (7)-(8) are equivalent to each other.

Formulation IP1

$$\min \quad \sum_{j\in F}\sum_{v\in V} w_j d_{jv} y_{jv} + \sum_{i\in C}\sum_{v\in V} u_i d_{iv} x_{iv} \tag{1}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{iv} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{v\in V} y_{jv} = 1 \qquad \forall j \in F \tag{3}$$

$$\sum_{j\in F} y_{jv} \geq x_{iv} \qquad \forall i \in C,\ v \in V \tag{4}$$

$$y_{jv}, x_{iv} \in \{0,1\} \qquad \forall i \in C,\ j \in F,\ v \in V \tag{5}$$

Formulation IP2

$$\min \quad \sum_{j\in F}\sum_{v\in V} w_j d_{jv} y_{jv} + \sum_{i\in C}\sum_{v\in V} u_i d_{iv} x_{iv} \tag{6}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{iv} = 1 \qquad \forall i \in C \tag{7}$$

$$\sum_{v\in V} y_{jv} = 1 \qquad \forall j \in F \tag{8}$$

$$\sum_{j\in F} y_{jv} = z_v \qquad \forall v \in V \tag{9}$$

$$x_{iv} \leq z_v \qquad \forall i \in C,\ v \in V \tag{10}$$

$$z_v, y_{jv}, x_{iv} \in \{0,1\} \qquad \forall i \in C,\ j \in F,\ v \in V \tag{11}$$

Model description

The IP formulations calculate an optimal solution to the MFLP by searching for the optimal destination vertices for the facilities and clients. For each facility j and client i, the optimal destination vertex v^j and v^i are searched respecitvely. By finding these vertices, the optimal solution of the MFLP is obtained. In Figure 2 an example of an improved solution of the MFLP is illustrated (Halper et al., 2015).

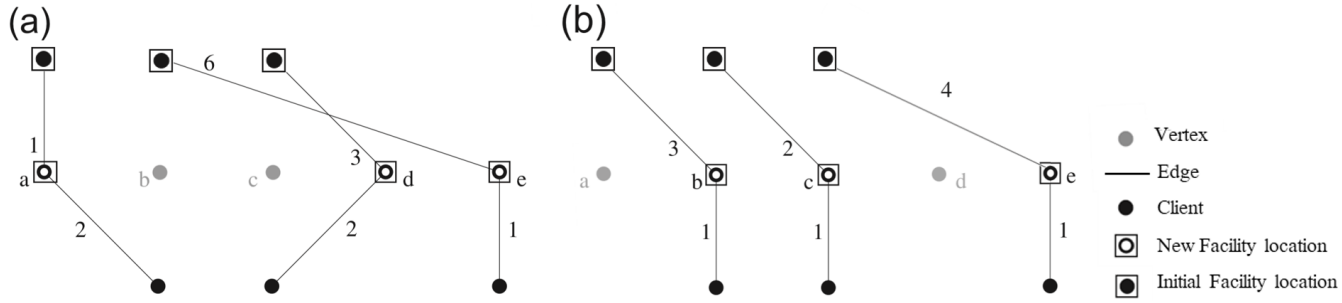

Figure 2: Example of a possible solution of the MFLP (Halper et al., 2015).

In Figure 2(a) the initial destinations of facilities $F \subseteq V$ at $\{a, d, e\}$ is displayed, together with the assignment of the clients to these facilities. In Figure 2(b) a relocation of two of the three facilities is displayed, from locations $\{\mathbf{a}, \mathbf{d}, e\}$ to $\{\mathbf{b}, \mathbf{c}, e\}$. By performing this relocation, the total sum of facility movement costs and client travel costs is decreased from 15 in Figure 2(a) to 12 in Figure 2(b). Hence, the relocation in Figure 2(b) can concluded to be an improvement of the facility allocation in Figure 2(a).

Model definition

The Objective functions (1) and (6) minimize the sum of the total weighted distances traveled by the facilities and clients together. Constraints (2) and (7) ensure that each client i has precisely one single destination vertex v. Equally, Constraints (3) and (8) ensure that each facility j has precisely one single destination vertex v. Furthermore, Constraints (4) make sure that client i can have vertex v as destination vertex if and only if the destination vertex v is a destination of some facility j. Constraints (9) are formulated such that $z_v = 1$, if vertex v is the destination of precisely one facility j. Otherwise, $z_v = 0$ and the vertex v is no destination of any facility j. Constraints (10) ensure that clients i can have vertex v as destination vertex if and only if the destination vertex v is a destination of some facility j, i.e. $z_v = 1$ for vertex v. Finally, Constraints (5) and (11) define the decision variables as binary variables.

Model assumptions

Without loss of generality, three assumptions are made for the two MFLP IP formulations of Halper et al. (2015). Formulation IP1 is based on the first two assumptions and formulation IP2 is based on all three assumptions. The model assumptions are as follows:

Assumption 1. *Each client is uniquely denoted by its vertex of origin $i \in C$.*
For two clients, having an identical vertex of origin $i \in C$, it can be assumed that they have the same destination vertex $v \in V$ in the optimal solution. If this is not the case, one client can be reassigned to the same destination vertex of the other, without increasing the objective function. Therefore, clients with an identical vertex of origin can be aggregrated into one single client with a weight equal to the sum of the individual weights of all aggregrated clients.

Assumption 2. *Each facility is uniquely denoted by its vertex of origin $j \in F$.*
For two facilities, it can be assumed that they have a different initial location $j \in F$. If this is not the case, one of the two facilities placed at position j is a copy j' of facility j, for $j' = j \in F$. Each client i, associated with one of the copies of the vertex j with cost of the edge (i, j'), can be reassigned to facility j with cost of edge (i, j), without increasing the objective function. Besides, no edges are created between copies of facilities (j, j'). Therefore, multiple facilities are not initially located at the same vertex.

Assumption 3. *There exists an optimal solution where no two facilities share the identical vertex of origin or vertex of destination.*
For the optimal solution, each client i should be assigned to the closest facility j location, i.e. d_{iv} is minimized. By Assumption 2, no facilities share the same vertex of origin. Hence, it may be assumed that no facilities will have the same vertex of destination in the optimal solution. Otherwise, one of the two facilities sharing the same destination, could stay alternatively at its original vertex v without increasing the objective value.

Model comparison

Table 9 provides a comparison between the two IP formulation of the MFLP, regarding the exact number of binary variables, variables, constraints and nonzero entries of the constraints matrix. From this comparison it can be concluded that IP2 has $|V|$ more variables and constraints compared to IP1. However, IP2 has a lower number of minimum necessary binary variables and consists of less nonzero coefficients in the constraint matrix than IP1. As a result, the IP2 enables larger instances to be solved in comparison to IP1.

Table 3: Comparison of the exact numbers of variables and constraints of the MFLP formulations IP1 and IP2, originally described by Halper et al. (2015).

Formulation	Binary variables	Variables	Constraints	Nonzero entries in constraint matrix
MFLP-IP1	$\|V\|\|F\|$	$\|V\|(\|C\|+\|F\|)$	$\|V\|\|C\|+\|C\|+\|F\|$	$\|V\|(\|C\|\|F\|+2\|C\|+\|F\|)$
MFLP-IP2	$\|V\|$	$\|V\|(\|C\|+\|F\|+1)$	$\|V\|\|C\|+\|V\|+\|C\|+\|F\|$	$\|V\|(3\|C\|+2\|F\|+1)$

4.2 LP Relaxation

By creating a linear programming (LP) relaxation of the IP formulations, lower bounds for the MFLP can be obtained. The LP relaxations for LP1 and LP2 are obtained by replacing the integrality constraints (5) and (11) by the following constraints (12) and (13), respectively:

$$y_{jv}, x_{iv} \geq 0 \qquad \forall i \in C,\ j \in F,\ v \in V \tag{12}$$

$$z_v, y_{jv}, x_{iv} \geq 0 \qquad \forall i \in C,\ j \in F,\ v \in V \tag{13}$$

4.3 Decomposition and new framework

In this subsection the decomposition of the MFLP into the facility assignment and client assignment subproblem is described. Besides, a new formulation framework for the MFLP is presented and illustrated.

Decomposition formulation

By supposing that a subset $Z \subset V$ containing p vertices is given, with the specification that all facilities f must have a vertex v in Z as destination vertex[2], the MFLP can be decomposed into the polynomially solvable *facility assignment (FA) subproblem* and the *client assignment (CA) subproblem*. Here, the decomposition of the MFLP is described, based on the IP2 formulation (Halper et al., 2015).

Facility assignment subproblem

The facility assignment (FA) subproblem is the first subproblem of the MFLP decomposition and can be introduced by the following definition.

Definition 2. *The facility assignment (FA) subproblem considers the bipartite matching between the p facilities $f \in F$ and the p corresponding destinations $v \in Z$, such that the total weighted bipartite matching cost is minimized.*

[2]Notice, this case is equivalent to fixing the variable z_v in IP2, i.e. $z_v = 1$ for all $v \in Z$ and $z_v = 0$ for all $v \in V \setminus Z$.

The size of this subproblem can be reduced by fixing the decision variable $y_{jv} = 0$ for each $j \in F$ and $v \in V \setminus Z$ in constraints (8). Furthermore, $z_v = 1$ for each $v \in Z$, such that constraints (9) can be rewritten as $\sum_{j \in F} y_{jv} = 1$ for each $v \in Z$. Based on these findings, the facility assignment subproblem is formulated, providing a least cost weighted bipartite matching between the initial and new facility destinations.

Formulation FA subproblem

$$FA(Z) = \min \quad \sum_{j \in F} \sum_{v \in Z} w_j d_{jv} y_{jv} \tag{14}$$

$$\text{s.t.} \quad \sum_{v \in Z} y_{jv} = 1 \qquad \forall j \in F \tag{15}$$

$$\sum_{j \in F} y_{jv} = 1 \qquad \forall v \in Z \tag{16}$$

$$y_{jv} \geq 0 \qquad \forall j \in F,\ v \in Z \tag{17}$$

The integrality constraints of y_{jv} can be relaxed since the constraint matrix of the FA subproblem is totally unimodular (Halper et al., 2015). Therefore, given a subset $Z \subset V$ containing p vertices, the $FA(Z)$ subproblem may be computed in polynomial time. For solving the FA problem, many existing algorithms can be used such as the Hungarian algorithm, introduced by Kuhn (1955).

Client assignment subproblem

The second subproblem of the MFLP decomposition is the client assignment (CA) subproblem. The definition of the CA subproblem is given here.

Definition 3. *The client assignment (CA) subproblem considers the assignment of each client $c \in C$ to a destination $v \in Z$, such that the total weighted travel distance of the clients is minimized.*

This subproblem can be created by fixing the decision variable $x_{iv} = 0$ for each $v \in V \setminus Z$ in constraints (7). Furthermore, constraints (10) can be rewritten as $x_{iv} \leq 1$ for each $i \in C, v \in Z$. Since the decision variables x_{iv} are binary, these constraints are redundant and can be left out. Based on these conclusions, the client assignment subproblem is formulated in which a destination in Z is chosen for each client $i \in C$ such that the weighted distance traveled by the clients is minimized.

Formulation CA subproblem

$$CA(Z) = \min \quad \sum_{i \in C} \sum_{v \in Z} u_i d_{iv} x_{iv} \tag{18}$$

$$\text{s.t.} \quad \sum_{v \in Z} x_{iv} = 1 \qquad \forall i \in C \tag{19}$$

$$x_{iv} \geq 0 \qquad \forall i \in C,\ v \in Z \tag{20}$$

The integrality constraints of x_{iv} can be relaxed since the constraint matrix of the CA subproblem is also totally unimodular (Halper et al., 2015). Since clients are assigned independent of each other, this subproblem can be further decomposed in a separate client assignment problem for each client. The client assignment could be solved for each client individually by computing all possible assignments and recording the one with minimum costs. By assigning each client with minimum costs, the CA subproblem is solved optimally.

Given the decomposition formulation and a subset $Z \subset V$ of p facilities, a unique solution can be found for the MFLP, based on the optimal solutions of both the FA and CA subproblem. In order to obtain optimal solutions for the MFLP, local search heuristics can be developed. These heuristics seek for the subset Z that provides a solution in which the total sum *FA(Z) + CA(Z)* is minimized (Halper et al., 2015). In Subsection 4.4 different local search heuristics will be provided.

New framework MFLP

Originally, the MFLP is formulated on a graph G(V,E) with subsets for the facilities $F \subseteq V$ and clients $C \subseteq V$. However, Halper et al. (2015) proposed a new framework for the MFLP, based on the decomposition of the MFLP into the facility and client assignment problems. In this new framework, the initial facility F and client C location sets are separated from the set of vertices V, such that the MFLP is formulated on a new graph $G(C \cup F \cup V, A)$, where C, F and V are disjoint sets. In Figure 3, an example is illustrated of a tranformation of the MFLP from the original (a) to the new described framework (b).

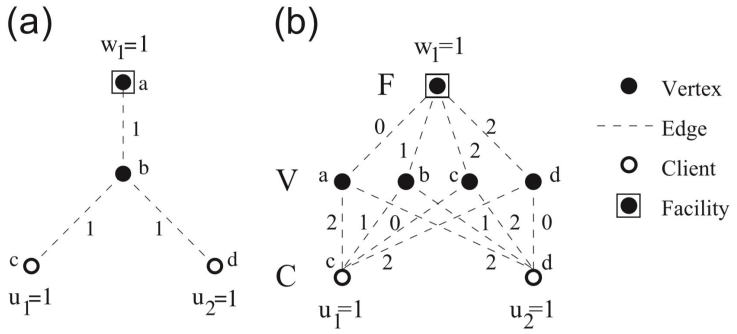

Figure 3: Example of a tranformation of the MFLP from the original (a) to the new described framework (b) (Halper et al., 2015).

4.4 Local search heuristics

Given the new decomposition formulation for the MFLP, local search heuristics can be effective for finding a subset Z which provides solutions in which the sum $FA(Z) + CA(Z)$ is minimized. One of the local search heuristics of Halper et al. (2015) is the n-OptSwap local search, described by the following definition .

Definition 4. *The n-OptSwap is a local search heuristic that searches for a neighborhood of solutions, consisting of all possible swaps between the $k \subset Z$ facility destinations of the current feasible solution and $k \subset V \setminus Z$ distinct alternative destinations, for $1 \leq k \leq n$. Each subset Z is associated with the solution found by solving the facility and client assignment subproblems optimally. If the new investigated solution improves the current solution, n-OptSwap continues looking for improvements in the neighborhood of the improved solution. Otherwise, the local search is finished and the final solution is obtained.*

Based on the n-OptSwap, the n-SmartSwap local search is developed (Halper et al., 2015). A definition for this local search is provided here.

Definition 5. *The n-SmartSwap is a hybrid local search heuristic that explores a neighborhood of solutions. This neighborhood consists of all possible swaps between the $k \subset Z$ current facility destinations and the $k \subset V \setminus Z$ distinct alternative destinations, for $1 \leq k \leq n$. For every possible swap in the neighborhood, a*

least cost perfect matching is solved. Finally, after no further improvements are found, the facility assignment subproblem is solved. If the investigated solution improves the current solution of the n-SmartSwap, the local search proceeds with looking for new improvements in the neighborhood of the improved solution. Otherwise, the final solution n-SmartSwap is obtained and the local search is finished.

For both the n-OptSwap and n-SmartSwap local search heuristics, two versions are considered. The first versions n-OptSwapFI and n-SmartSwapFI improve the solution by performing the first improvement (FI) found. Alternatively, the second versions n-OptSwapBI and n-SmartSwapBI perform the best improvement (BI) found in the complete neighborhood.

From the two defined local search heuristics, the hybrid local search n-SmartSwap has been concluded to be the best heuristic for quickly computing high quality solutions (Halper et al., 2015). Therefore, the high quality and quickly computed MFLP solutions of the n-SmartSwap will be considered as the baseline in this thesis. The pseudocode for the n-SmartSwap local search is presented in Appendix A.

In Figure 4, an example is illustrated of the 2-SmartSwap local search that generates a MFLP solution (Halper et al., 2015). Figure 4(a) displays the initial solution with $Z = \{a, d, e\}$. In Figure 4(b) the least cost perfect matching problem between the two facilities identified and the vertices b and c is illustrated. In this panel, the replacement is displayed of facilities a and d in Z with respectively b and c in $V \setminus Z$. Figure 4(c) displays the neighborhood solution found by solving this problem, with a cost of 14. The 2-SmartSwap local search constructs a solution by iteratively exploring all different neighborhood solutions and selecting the one with the least cost.

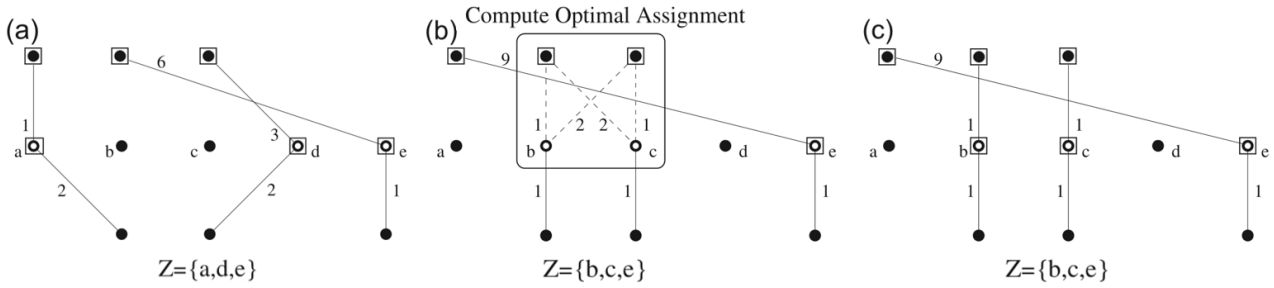

Figure 4: Example of a generated solution by the 2-SmartSwap local search (Halper et al., 2015).

4.5 ForwardSwap meta-heuristic

Because the MFLP is $\mathcal{NP}$-hard (Garey and Johnson, 1979), the challenge remains to find efficient algorithms for this combinatorial optimization problem (Voßs et al., 2005). Therefore, a new meta-heuristic is developed and described in this subsection. Besides, different time restrictions of the meta-heuristic are explained, controlling the computation times of this algorithm.

ForwardSwap description

From the local search measure variants, the FI is focused on an immediate (first) improvement and the BI is focused on a somewhat delayed (best) improvement. By focusing on (almost) immediate improvements, both heuristics could possible fall into the *greedy trap* of making shortsighted suboptimal choices.

Therefore, in this thesis the new meta-heuristic ForwardSwap is developed for the MFLP. The ForwardSwap has incorporated intelligent look ahead mechanisms within the improvement measure evaluation. This

new improvement measure is focused on the most promising solution instead of a shortsighted immediate improvement and is a modification of the *pilot method* of Duin and Voß (1999) (described in Section 3.3). The pilot method is a meta-heuristic incorporating a modification of a subordinate heuristic.

The ForwardSwap searches the best final solution by iteratively solving the 1-SmartSwapBI local search for every possible swap and choosing the swap with the most promising final solution. By means of this look ahead mechanism, the greedy swaps of the SmartSwap local search heuristic is tempered and the final MFLP solutions possibly improve. The ForwardSwap can be described by the following definition.

Definition 6. *The ForwardSwap is a meta-heuristic iteratively building up a high-quality master solution. For every iteration, this master process performs two steps. Firstly, a pilot solution is computed for every swap of Z that results in an improvement of the current solution*[3]*, where the pilot solution consists of performing the 1-SmartSwapBI. Secondly, the swap with the best pilot solution (lowest cost) is recorded. After computing all pilot solutions in the iteration, the current solution is updated by the swap with the best pilot solution if this improves the current ForwardSwap solution. In this case, the master solution is updated by the best pilot solution if and only if this results in an improvement. Furthermore, the meta-heuristic proceeds with looking for new improvements in the neighborhood of the improved solution. However, if the best pilot solution does not improve the current ForwardSwap solution, the ForwardSwap is finished and the master solution is returned.*

The described procedure of the ForwardSwap is illustrated in a chart diagram in Figure 5. This figure provides among others a visualization of the interaction between the iterating master process and the updates of the current solution and master solution.

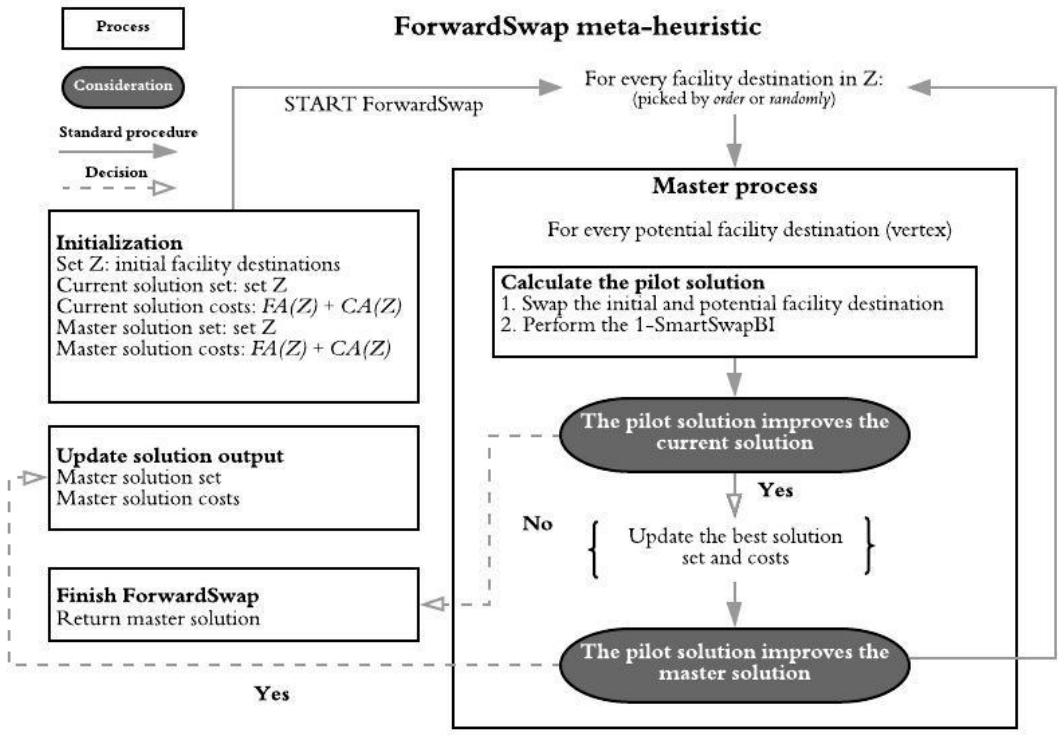

Figure 5: Chart diagram of the procedure of the meta-heuristic ForwardSwap.

[3]This condition is based on the trade-off made between the decreasing solution quality and the drastically improving runtime.

ForwardSwap time restrictions

In Section 6 the results of the ForwardSwap are provided for different instances. In accordance to the research of Voßs et al. (2005), the modified pilot method ForwardSwap appears to be computationally time expensive for large instances. Therefore, the implementation of the ForwardSwap is extended with time restrictions.

Three different versions of the ForwardSwap are implemented, all containing three different time restriction parameters: one alternating parameter for restricting the individual iteration time and two fixed parameters for restricting the overall running time of the ForwardSwap.

Iteration time restriction

The *iteration time restriction* restricts the search time for improvements for each individual facility, i.e. after this time the heuristic proceeds with looking for possible improvements by considering swaps of the next facility destination in Z. Every time the time parameter is exceeded, the local search finishes the current iteration. Afterwards, the running time is initialized to zero and the meta-heuristic proceeds by searching for swaps with the next facility destination in Z. The time parameters is denoted between brackets after the local search name. For making motivated choices for the time parameter values, the ForwardSwap is performed for multiple values. From these tests, values between the 0.001 and 0.1 seconds generally provided the best solutions. Therefore, in this thesis the following three time parameters are considered, denoted in seconds: $\{0.1\}$, $\{0.02\}$ and $\{0.001\}$.

Execution time restrictions

Because the sum of a large number of time restricted iterations still can result in long running times, two fixed parameters are implemented for restricting the total running time of the ForwardSwap. In the implementation for the ForwardSwap, the total running time is restricted by the *overall time restriction* of 900 seconds. This restriction ensures that if the running time exceeds the 900 seconds, the meta-heuristic finishes the current iteration and returns the master solution currently found as output.

Finally, an extra incentive for reducing the running time of the ForwardSwap is created by implementing the *master time restriction*. If a new solution is found, this implemented restriction checks if the pilot solution improves the current master solution. If no master solution is found for longer than 300 seconds, the meta-heuristic finishes the current iteration and returns the master solution found so far. The restriction parameter of 300 seconds prevents the heuristic from searching for improvements that are not existing or reachable in a restricted period of time.

Both execution time restrictions are chosen such that solutions are provided within about 900 seconds. However, these time limitations can be adjusted to any time limit desired. Due to the implemented time restrictions, the number of possible improvements considered within every iteration is restricted. However, this is an inevitable consequence of making the trade-off between solution quality and computation time.

Facility swap order

The standard ForwardSwap procedure forces the meta-heuristic to search for improvements in an ordered way, i.e. starting with investigating swaps with the first facility destination in set Z and proceeding with the subsequent facilities. However, in this way the ForwardSwap may only get the possibility to investigate the first few facilities in set Z for every iteration, which could lead to suboptimal solution outcomes. Hence, two different types of the ForwardSwap are considered. The *Forward Swap Ordered (ForwardSwapO)* follows the standard Forward Swap procedure, while the *Forward Swap Random (ForwardSwapR)* investigates a random facility destination in Z in every iteration.

5 Data Description

In this thesis, data instances are used from the database provided by Raghavan (2019). This database contains the data instances used in the research of Halper et al. (2015). In this way, the research of Halper et al. (2015) can be reproduced. The database of Raghavan (2019) consist of instances of homogeneous and heterogeneous facilities; respectively facilities with equal and different facility weights. For both types, instances are provided with small and large *client/facilities ratios* $|C|/|F|$'s. This characteristic represents the ratio of client weights to facility weights, such that for small and large $|C|/|F|$'s the incentive to relocate facilities rather than moving clients is low and high, respectively. In Table 4 an overview is presented of the homogeneous instances for small and large $|C|/|F|$'s (Raghavan, 2019).

Table 4: Features of homogeneous data instances with small and large $|C|/|F|$'s.

Instances with a large $\|C\|/\|F\|$				Instances with a small $\|C\|/\|F\|$			
Clients	Instances	Avg. facilities	Avg.$\|C\|/\|F\|$	Clients	Instances	Avg. facilities	Avg. $\|C\|/\|F\|$
100	1	5	20.0	100	4	18.3	7.0
200	2	7.5	30.0	200	3	42.3	6.0
300	2	7.5	45.0	300	3	63.3	6.0
400	2	7.5	60.0	400	3	84.3	6.0
500	2	7.5	75.0	500	3	105.7	6.0
600	2	7.5	90.0	600	3	126.7	6.0
700	2	7.5	105.0	700	1	70.0	10.0
800	2	7.5	120.0				
900	2	7.5	135.0				
Avg.	1.9	7.2	75.6	Avg.	2.9	72.9	6.7

The scope of this thesis is on the instances with a small $|C|/|F|$. In this thesis, instances with a number of vertices beneath the 300 are referred to as small instances. Logically, instances with a number of vertices greater or equal to 300 are considered as large instances.

6 Results

In this section, results are provided for the IP formulation, the two implemented local search heuristics n-OptSwap and n-SmartSwap for variants = $\{FI, BI\}$ and $n = \{1, 2\}$ and for different variants of the meta-heuristic ForwardSwap. For all large instances, the overall time restriction of 900 seconds and the master time restriction of 300 seconds are implemented.

6.1 Replication

In this subsection, the replicated results of the research of Halper et al. (2015) are provided. Firstly, the results of the IP1, IP2 and LP2 formulations are presented. Secondly, the results of the n-OptSwap and n-SmartSwap formulations are discussed for $n = \{1, 2\}$ for both measures FI and BI. Additionally, a discussion is provided in which the replicated results are compared to the results of Halper et al. (2015).

IP and LP formulations

Table 5 presents the number of locations, facilities and clients for all 20 homogeneous problem instances with small $|C|/|F|$'s (Raghavan et al., 2019). The MFLP is solved for all these problem instances using the IP1, IP2 and LP2 formulation. The results are also presented in Table 5. Finally, all running times (RT) are provided and denoted in seconds.

The computational results indicate that the solving times of the IP1 are on average more than 4 times longer than the running times of the IP2, as the average running time of the IP1 is 56.12 seconds instead of 12.78 seconds for the IP2. According to these results, the IP2 formulation can be concluded to be a better formulation than the IP1. However, both IP formulations provide optimal solutions within a reasonable computation time of on average less than one minute. Furthermore, the IP-LP gaps are calculated based on the IP2 and the results of the LP2 model. The LP2 relaxation has produced integer valued solutions in 18 of the 20 problem instances, yielding optimal solutions. Consequently, the IP-LP gaps are extremely close to 0.00%. This indicates that the LP relaxation provides lower bounds of high quality for the MFLP.

Table 5: Information, results and comparisons of the MFLP for 20 instances, using the IP1, IP2 and LP2.

Instance	Locations	Facilities	Clients	IP1	IP1 RT(s)	IP2	IP2 RT(s)	LP2	LP2 RT(s)	IL-LP Gap (%)
2	100	10	100	4914	1.05	4914	0.51	4914	0.29	0.00
3	100	10	100	5792	0.77	5792	0.31	5792	0.19	0.00
4	100	20	100	4748	0.89	4748	0.33	4748	0.34	0.00
5	100	33	100	2170	1.16	2170	0.42	2170	0.17	0.00
8	200	20	200	6077	3.42	6077	2.52	6077	3.01	0.00
9	200	40	200	4348	4.06	4348	2.05	4348	1.07	0.00
10	200	67	200	2377	5.71	2377	1.46	2377	1.05	0.00
13	300	30	300	5487	14.55	5487	12.86	5486	7.90	0.02
14	300	60	300	3963	19.86	3963	8.00	3963	5.54	0.00
15	300	100	300	2642	16.74	2642	5.51	2642	3.95	0.00
18	400	40	400	5844	30.41	5844	20.32	5844	14.41	0.00
19	400	80	400	4229	44.87	4229	22.98	4229	11.35	0.00
20	400	133	400	3178	65.09	3178	12.63	3178	8.69	0.00
23	500	50	500	6756	76.59	6756	18.88	6756	35.80	0.00
24	500	100	500	4782	88.41	4782	16.38	4782	18.90	0.00
25	500	167	500	3033	153.26	3033	10.56	3033	5.98	0.00
28	600	60	600	6133	102.67	6133	40.31	6131	30.28	0.03
29	600	120	600	4756	138.14	4756	24.37	4756	15.09	0.00
30	600	200	600	3151	205.98	3151	16.25	3151	12.77	0.00
33	700	70	700	6740	148.84	6740	39.04	6740	64.50	0.00
Avg.					56.12		12.78		12.06	0.00
Max.					205.98		40.31		64.50	0.03

The computational running times of the replicated results in Table 5 are on average larger than the running times of the original results of Halper et al. (2015)[4]. For the homogeneous instances, the average running times of the replicated IP1 results are approximately 17% higher than the results of Halper et al. (2015), with 56.12 seconds instead of 47.82 seconds. Equally for IP2, there is a slight increase in running time of approximately 5%, from 12.15 seconds of Halper et al. (2015) to 12.78 seconds of the replicated IP2 formulation. This difference can be caused by a more efficient implementation of the IP formulations in the research of Halper et al. (2015).

[4]In Appendix B the computational results of Halper et al. (2015) are provided for formulations IP1, IP2 and the local search heuristics n-OptSwap and n-SmartSwap for the 20 homogeneous instances.

1-Swap local search heuristics

For 10 small and 5 large problem instances, Table 6 presents the computational results provided by the 1-OptSwap and 1-SmartSwap for both variants *FI* and *BI*. The results in Table 6 indicate that the local search heuristics produce solutions of high quality for small problem instances, i.e. small locality gaps close to 0.00%. However, because the 1-OptSwapFI, 1-OptSwapBI and 1-SmartSwapFI are computationally time expensive, the solution procedures had to be cut off for large instances. Consequently, the solutions of these problem instances are of low quality, i.e. large locality gaps between the 10.00% and 25.00%. As a result, the 1-SmartSwapBI outperforms the other three 1-swap local search algorithms 1-OptSwapFI, 1-OptSwapBI and 1-SmartSwapFI with an average locality gap of 0.90% instead of 4.98%, 4.98% and 6.37% and an average runtime of 274.08 seconds instead of 447.71 seconds, 470.90 seconds and 431.14 seconds respectively. From these results the n-SmartSwap can be confirmed to be the best local search heuristic for quickly computing high quality solutions (Halper et al., 2015).

Table 6: Results of the MFLP using 1-OptSwap and 1-SmartSwap for variants = {*FI*, *BI*}.

Instance	1-OptSwapFI		1-OptSwapBI		1-SmartSwapFI		1-SmartSwapBI		IP2
	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Runtime(s)
2	0.00	0.42	0.00	0.33	0.43	0.02	0.00	0.01	0.45
3	0.00	0.30	0.00	0.28	1.28	0.03	0.00	0.06	0.46
4	0.16	3.20	0.32	1.70	2.60	0.22	1.30	0.16	0.42
5	0.00	3.80	0.58	2.83	5.04	0.40	0.58	0.25	0.18
8	0.00	10.71	0.03	5.82	1.58	0.39	0.03	0.12	2.16
9	0.00	68.94	0.24	142.92	1.23	5.49	0.08	0.18	1.78
10	0.45	253.57	0.22	495.86	3.46	30.62	1.60	8.20	1.19
13	0.02	65.52	0.02	99.01	0.43	114.89	0.38	7.15	10.06
14	1.35	900.12	0.63	900.47	2.91	900.26	0.88	35.48	4.25
15	3.84	900.37	2.04	900.74	4.35	900.58	0.00	59.86	2.57
25	12.97	901.95	12.72	903.28	11.22	905.80	2.40	900.25	13.45
28	13.54	902.02	12.32	901.91	14.19	900.53	0.39	233.14	61.30
29	12.27	900.31	12.66	902.03	12.98	901.00	1.26	1065.83	23.55
30	11.55	902.41	14.47	903.80	12.78	905.52	3.74	900.42	14.88
33	18.51	901.96	18.50	902.49	21.02	901.42	0.91	900.15	51.34
Avg.	4.98	447.71	4.98	470.90	6.37	431.14	0.90	274.08	12.54
Max.	18.51	902.41	18.50	903.80	21.02	905.80	3.74	1065.83	61.30

The replicated solution outcomes of the hybrid 1-SmartSwapBI local search in Table 6 have an average running time of 274.08 seconds. This is more than 20 times longer than the average running time 12.78 seconds of the replicated IP2 and even more than 60 times longer than the average original running time 4.29 seconds of the implemented 1-SmartSwapBI of (Halper et al., 2015). These extremely large differences in runtime could be explained by the efficiency of the local search implementations of Halper et al. (2015), missing in the implementation of the replication in this thesis.

Regarding the solution quality, the replicated solutions have an average locality gap of 0.90%. Compared to the average locality gap 0.66% of the research of Halper et al. (2015), the replicated average locality gap is relatively more than 35% larger. However, this difference in average locality gap can be explained as a consequence of the large running times. Because of the relatively inefficient implementation of the replication and the implemented time restrictions, some 1-SmartSwapBI procedures are cut off. By cutting off solution procedures, solutions with large locality gaps may be obtained, declaring the larger average locality gap of the replication part.

In contrast, for small instances the replicated solutions are of high quality. In fact, for some instances the locality gaps of the replication are even smaller than the original locality gaps, e.g. the replicated solutions of instances 10 and 13 have locality gaps of respectively 0.08% and 0.38% instead of 0.63% and 1.22%.

Because the conclusions for the 1-SmartSwapBI are representative for all 1-swap local searches, the 1-swap local search replications can concluded to be implemented inefficiently in general. Consequently, the 1-swap local searches only provide high quality solutions for small instances. For large instances, the solution quality declines due to the implemented time restrictions.

2-Swap local search heuristics

In Table 7 the computational results of 5 small and 4 large problem instances are presented, provided by performing the 2-OptSwap and 2-SmartSwap for both variant *FI* and *BI*. Besides, these results are compared with the computational results of the new meta-heuristic ForwardSwap. Notice, the time restrictions are not applied to the implementation of the original ForwardSwap.

The computational results in Table 6 describe solutions of very high quality for small instances, i.e. locality gaps close to 0.00%. For small instances, the 2-OptSwapFI and the meta-heuristic ForwardSwap are providing the solutions of highest quality with locality gaps of only 0.00%, except for one locality gap of ForwardSwap which is equal to 0.11%. However, even for the small instances the runtimes are already extending to undesired running times, e.g. a minimum runtime of 436.40 seconds of all computed solutions of the 8^{th} instance. Because the iterations of these heuristics are computational expensive, sometimes not even one improvement has been found for large instances, resulting in large locality gaps between the 10% and 30%. As a result, the 2-swap local search heuristics and the ForwardSwap can concluded to be inappropriate for solving large problem instances.

Table 7: Results of the MFLP using 2-OptSwap, 2-SmartSwap (for variants = {*FI*, *BI*}) and ForwardSwap.

Instance	2-OptSwapFI		2-OptSwapBI		2-SmartSwapFI		2-SmartSwapBI		ForwardSwap	
	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)
2	0.00	6.72	0.00	35.76	0.04	19.84	0.00	9.82	0.00	8.95
3	0.00	5.82	0.00	27.07	0.00	15.86	0.00	7.31	0.00	19.58
4	0.00	94.09	0.32	581.59	0.00	282.32	1.30	34.90	0.11	86.93
5	0.00	348.59	0.58	1765.48	0.23	711.76	0.58	102.54	0.00	107.43
8	0.00	436.40	0.03	2982.07	0.27	1793.86	0.03	588.49	0.00	559.69
25	12.98	2668.78	15.16	984.94	10.57	901.88	15.16	904.88	2.53	4288.65
28	15.71	900.31	29.97	923.67	13.08	900.39	29.97	906.93	0.36	1082.27
29	13.08	900.39	20.02	1153.69	12.98	900.78	20.02	900.63	0.90	2810.35
30	11.89	904.58	18.37	2042.43	13.58	903.34	18.37	927.33	3.00	115725.69
Avg.	5.96	696.19	9.38	1166.30	5.64	714.45	9.49	486.98	0.77	13854.39
Max.	15.71	2668.78	29.97	2982.07	13.58	1793.86	29.97	927.33	3.00	115725.69

For making a comparison between the presented replicated results in Table 7 and the original results of Halper et al. (2015), the 1-SmartSwapBI should have been performed before performing the 2-swap heuristics. In this case, the solution obtained could be used as an *educated initial solution*, i.e. a solution expected to be relatively close to the optimal solution. However, by first performing the 1-SmartSwapBI, the restricted time of 900 seconds would already had been exceeded for large instances, resulting in the same results obtained by the 1-SmartSwapBI in Table 6. Therefore, the 2-swap local search heuristics are performed from the initial facility destinations and no comparison can be made.

6.2 ForwardSwap meta-heuristic

Table 8 presents the computational results of 10 small and 5 large problem instances for different modifications of the original ForwardSwap. Results of the ForwardSwap are provided for both *ordered (O)* and *random (R)* facility swaps and for the iteration time restrictions {0.1}, {0.02} and {0.001}. Additionally, the execution time, iteration time and master time restrictions are all implemented for all variants of the ForwardSwap.

In Table 8 the results of the six variants of the ForwardSwap are presented. From the results no universal best meta-heuristic is indicated. Comparing the *ordered* and *random* facility swaps, the average locality gaps of the ForwardSwapO(0.1) and ForwardSwapO(0.02) respectively 0.69% and 0.69% are larger than the locality gaps 0.46% and 0.49% of ForwardSwapR(0.1) and ForwardSwapR(0.02) respectively. However, the locality gap 0.58% of ForwardSwapO(0.001) is lower than the locality gap 0.60% of ForwardSwapR(0.001). As a result, no facility swap variant strictly outperforms the other variant.

Regarding the iteration times, a decrease in iteration time is associated with a decrease in the average running time for all ForwardSwap variants. Nevertheless, for some specific instances this association does not hold, e.g. for the 13^{th} instance the ForwardSwapR(0.02) has a running time of 573.36 seconds while the ForwardSwapR(0.001) has a running time of only 380.31 seconds. This can be explained by the insight that earlier cut offs of iterations can result in sub-optimal improvements, possibly requiring more iterations to obtain a high quality solution. This can result in longer running times for meta-heuristics with smaller iteration time restrictiosn, compared to meta-heuristics with larger time restrictions.

From the six ForwardSwap meta-heuristics, three ForwardSwap variants outperform the other meta-heuristics, referring to as the candidates for the best solution method for the MFLP. From all ForwardSwapO heuristics, the ForwardSwapO(0.001) can concluded to be the best meta-heuristic and first candidate, with an average locality gap of 0.58% instead of 0.69% and 0.69% and an average running time of 398.48 seconds instead of 495.96 seconds and 495.15 seconds for ForwardSwapO(0.1) and ForwardSwapO(0.02) respectively. Compared to the ForwardSwapO(0.001), the ForwardSwapR(0.001) is outperformed, with on average a larger locality gap of 0.60% and a larger average running time of 459.12 seconds. Consequently, the ForwardSwapR(0.1) and ForwardSwapR(0.02) are the other two candidates, with average locality gaps of 0.46% and 0.49% and average computation times of 495.19 seconds and 477.83 respectively.

Table 8: Results of the ForwardSwap for versions 0.1, 0.02 and 0.001 for ordered and random facility swaps.

Instance	ForwardSwapO(0.1)		ForwardSwapO(0.02)		ForwardSwapO(0.001)		ForwardSwapR(0.1)		ForwardSwapR(0.02)		ForwardSwapR(0.001)	
	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)
2	0.00	0.43	0.00	0.35	0.00	0.25	0.00	2.75	0.00	2.80	0.00	2.27
3	0.00	2.33	0.00	2.10	0.00	2.30	0.00	3.78	0.00	3.77	0.00	1.54
4	0.11	6.81	0.11	6.95	0.11	7.08	0.00	30.68	0.32	23.74	0.11	14.34
5	0.58	22.53	0.58	17.56	0.58	16.11	0.00	32.73	0.58	16.05	0.75	13.55
8	0.03	51.35	0.03	51.21	0.03	51.45	0.03	99.20	0.03	104.25	0.03	75.12
9	0.06	362.06	0.06	361.63	0.06	362.16	0.36	288.49	0.07	208.02	0.29	120.02
10	0.96	328.70	0.96	328.94	1.13	60.18	0.66	207.76	1.15	218.79	1.53	138.57
13	0.38	317.00	0.38	316.06	0.24	473.05	0.00	807.23	0.00	573.36	0.07	380.31
14	0.33	900.02	0.33	900.77	0.56	253.38	0.71	533.70	0.33	574.71	0.52	833.95
15	0.00	900.66	0.00	901.29	0.22	442.33	0.00	900.08	0.00	900.08	0.00	789.51
25	0.97	904.17	0.97	902.86	0.89	654.37	0.90	909.62	0.82	909.46	0.89	910.24
28	0.37	908.31	0.47	902.79	0.39	903.12	0.39	901.55	0.18	900.41	0.26	900.84
29	3.91	920.19	3.91	916.09	1.95	924.51	0.66	907.95	0.96	904.35	0.88	902.14
30	2.13	914.67	1.93	911.86	1.93	922.79	2.36	900.49	2.36	925.66	3.05	903.90
33	0.56	900.19	0.56	906.84	0.56	904.12	0.83	901.88	0.58	902.01	0.55	900.56
Avg.	0.69	495.96	0.69	495.15	0.58	398.48	0.46	495.19	0.49	477.83	0.60	459.12
Max.	3.91	920.19	3.91	916.09	1.95	924.51	2.36	909.62	2.36	925.66	3.05	910.24

6.3 Best solution approach

For determining the best solution method for providing high quality MFLP solutions within short computation times, four candidate solution methods are considered an compared. These four candidates consist of the three meta-heuristics ForwardSwapO(0.001), ForwardSwapR(0.1) and ForwardSwapR(0.02) and the hybrid 1-SmartSwapBI local search heuristic. In Table 9 the results of the four candidates are presented. From these results, the inevitable trade-off between solution quality and computation time had to be made. Based on this trade-off, three possible points of view are discussed.

Table 9: Results of three variants of the ForwardSwap and the 1-SmartSwapBI.

	ForwardSwapO(0.001)		ForwardSwapR(0.1)		ForwardSwapR(0.02)		1-SmartSwapBI	
	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)
Avg.	0.58	398.48	0.46	495.19	0.49	477.83	0.90	274.08
Max	1.95	924.51	2.36	909.62	2.36	925.66	3.74	1065.83

Focusing on the highest solution quality, the ForwardSwapR(0.1) is the best solution method, providing solutions with the smallest locality gap of on average 0.46%. Comparing this meta-heuristic to the 1-SmartSwapBI, the locality gap of the 1-SmartSwapBI can be relatively decreased by almost 50% from 0.90% to 0.46%. However, this improvement is in exchange for a relatively increase of the running time with more than 80% from 274.08 seconds to 495.19 seconds.

Alternatively, focusing on the computational running time, the hybrid 1-SmartSwapBI local search remains the best solution method for quickly computing solutions of reasonable high qualities with an average locality gap of 0.90% computed within a running time of on average 274.08 seconds.

Finally, by making an equal trade-off between solution quality and computation time, the meta-heuristic ForwardSwapO(0.001) could be concluded to be the new best solution method for the MFLP. Comparing this meta-heuristic to the 1-SmartSwapBI, the solution quality of the ForwardSwapO(0.001) is improved by a relatively decrease of the average locality gap of approximately 35% from 0.90% to 0.58% in exchange for a relatively increase in running time of 26% from 274.08 seconds to 398.48 seconds.

Although the ForwardSwapR(0.1) and ForwardSwapR(0.02) have provided higher quality solutions, these solutions are based on a random order. Consequently, these meta-heuristics cannot guarantee solutions with stable locality gaps and running times. Moreover, the solution outcomes of the ForwardSwapO(0.001) did not exceed a deviation of more than 1.95% from the optimal solution, compared to the maximum deviations 2.36%, 2.36% and 3.74% of the other three solution methods. From all these results, the ForwardSwapO(0.001) can concluded to be the most stable and robust solution method for computing high quality solutions of the MFLP within short computation times.

7 Conclusion

Because the MFLP is $\mathcal{NP}$-hard, computational running times of IP formulations for the MFLP grow drastically by increasing problem instance sizes. Therefore, multiple local search heuristics are developed for the MFLP. From the results, the hybrid local search is confirmed to be the best current heuristic for quickly computing high quality solutions for the MFLP, combining the speed and solution quality of two other implemented local search heuristics. However, the challenge to find efficient algorithms for the combinatorial optimization problem MFLP remains.

Therefore, in this thesis a new meta-heuristic is developed. This solution approach is a master process consisting of intelligent look ahead mechanisms. In this algorithm, the meta-heuristic searches the most promising step by performing the hybrid local search for every possible step. By iteratively choosing the step with the best final solution, the meta-heuristic builds up a master solution based on look ahead results. In this thesis, six different variants of the meta-heuristic are developed and performed for different problem instances.

For the development of an algorithm that provides high quality solutions for the MFLP within short computational running times, the inevitable trade-off between solution quality and computation time has to be considered. From this trade-off, one variant of the new meta-heuristic has concluded to be the new best solution method for the MFLP. In this meta-heuristic, an iteration time restriction of 0.001 seconds is incorporated and swaps of facilities are considered in a fixed order.

The new developed meta-heuristic has provided reproducible high quality solution outcomes with less than 2% deviation from the optimal solution, provided within relatively short computation times of on average 7 minutes. Specifically, this meta-heuristic provides an improvement of the solution quality of the 1-SmartSwapBI local search with a relatively decrease of the average locality gap of approximately 35% from 0.90% to 0.58% in exchange for a relatively increase in computation time of 26% from 274.08 seconds to 398.48 seconds. Based on these results, this new meta-heuristic can concluded to be the best solution method for providing companies and organizations high quality MFLP solutions within short computation times.

However, much further research could be done to the MFLP, focusing on improving the developed meta-heuristic in this thesis. In the context of companies and organizations, the trade-off between the solution quality and computation time could be reconsidered for companies individually. In this way, solution methods could be made more efficient and appropriate to the unique context of companies and organizations.

Furthermore, the possible differences in effectiveness of the meta-heuristic could be considered for instances with small and large client versus facility ratios. Besides, the different restriction parameters could be investigated in more detail, performing multiple test runs in order to find more efficient restriction parameter settings. Alternatively, the time restrictions could be replaced by implementing a restrictive evaluation depth, which refers to the number of times the meta-heuristic restarts the heuristic procedure. Another possible improvement that could be investigated is the improvement of the initial solution of the meta-heuristic by solving one of the local search heuristics before performing the meta-heuristic.

Additionally, further research could be done to implement the local search heuristics and meta-heuristic more efficiently. In this way, running times can be controlled which enables the meta-heuristic to compute more iterations in which solutions of higher quality may be found. Furthermore, new mechanisms for the meta-heuristic could be developed and incorporated in the current implementation. Finally, further detailed research to the meta-heuristic in this thesis could possibly contribute to the development of algorithms that provide solutions for the MFLP of even higher quality within shorter computation times.

References

Bespamyatnikh, S., Bhattacharya, B., Kirkpatrick, D., and Segal, M. (2000). Mobile facility location. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 46–53.

Cornuéjols, G., Nemhauser, G., and Wolsey, L. (1983). The uncapicitated facility location problem. Technical report, Cornell University Operations Research and Industrial Engineering.

Demaine, E. D., Hajiaghayi, M., Mahini, H., Sayedi-Roshkhar, A. S., Oveisgharan, S., and Zadimoghaddam, M. (2009). Minimizing movement. *ACM Transactions on Algorithms (TALG)*, 5(3):1–30.

Duin, C. and Voß, S. (1999). The pilot method: A strategy for heuristic repetition with application to the steiner problem in graphs. *Networks: An International Journal*, 34(3):181–191.

Friggstad, Z. and Salavatipour, M. R. (2011). Minimizing movement in mobile facility location problems. *ACM Transactions on Algorithms (TALG)*, 7(3):1–22.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability*, volume 174. freeman San Francisco.

Halper, R., Raghavan, S., and Sahin, M. (2015). Local search heuristics for the mobile facility location problem. *Computers & Operations Research*, 62:210–223.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Meyerson, A. (2001). Profit-earning facility location. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 30–36.

Pearl, J. (1984). Intelligent search strategies for computer problem solving. *Addision Wesley*.

Raghavan, S. (2019). Instances for the capacitated mobile facility location problem. data retrieved from Raghavan, S., https://doi.org/10.17632/JTV74HKMFG.1.

Raghavan, S., Sahin, M., and Salman, F. S. (2019). The capacitated mobile facility location problem. *European Journal of Operational Research*, 277(2):507–520.

Rayward-Smith, V. J. and Clare, A. (1986). On finding steiner vertices. *Networks*, 16(3):283–294.

Voßs, S., Fink, A., and Duin, C. (2005). Looking ahead with the pilot method. *Annals of Operations Research*, 136(1):285–302.

8 Appendix

Appendix A

In this appendix, a pseudocode is provided for the n-SmartSwap local search heuristic.

Pseudocode 1: n-SmartSwap Local Search

```
The n-SmartSwap local search heuristic explores a neighborhood of solutions, searching for the optimal
 set of facility destination Z.
Result: Optimal solution MFLP
Input: Sets and Parameters;
FACILITIES (F), list of current facility locations;
CLIENTS (C), list of current client locations;
VERTICES (V), list of potential destination locations for the facilities;
DISTANCES, matrix consisting of the distances between all vertices in F, C and V;
WEIGHTS FOR F AND C, list of individual weights of the facilities and clients;

Initialization:
Z = FACILITIES, set of new facility destinations;
BEST_SOLUTION = Z;
BEST_SOLUTION-costs = FA(Z)-costs + CA(Z)-costs;

Updating and improving set Z: exploring neighborhood solutions
Z = n-SmartSwap_Iteration(Z):
getAllSubsets(Z, V);
if First Improvement then
    for facilities in Z do
        for vertices in V\Z do
            HungarianMethod(): { Facility_Assignment, FA_Hungarian-costs }
            CANDIDATE_SOLUTION = SwapMethod(Facility_Assignment);
            CANDIDATE-costs = FA_Hungarian-costs + CA(CANDIDATE_SOLUTION)-costs;

            if CANDIDATE-costs ≤ BEST_SOLUTION-costs then
                Z = CANDIDATE_SOLUTION;
                BEST_SOLUTION-costs = CANDIDATE-costs ;
                return n-SmartSwap_Iteration(Z);
            end
        end
    end
    return Z
end

Update BEST_SOLUTION = Z;
BEST_SOLUTION-costs = FA(Z)-costs + CA(Z)-costs;

Optimality check:
OPTIMALITY_CHECK-costs = FA(Z)-costs + CA(Z)-costs;
if CURRENT_SOLUTION-costs ≤ OPTIMALITY_CHECK-costs then
    return New Facility Destinations = Z; New Client Destinations = CA(Z);
end
else
    Update:
    BEST_SOLUTION = CA_subproblem(Z);
    BEST_SOLUTION-costs
end
```

Appendix B

In this appendix, computational results of Halper et al. (2015) are provided. In Table 10 the solutions are provided of the formulations IP1, IP2 and in Figure 11 the solutions are provided of the local search heuristics n-OptSwap and n-SmartSwap, both for 20 homogeneous instances (Raghavan et al., 2019).

Table 10: MFLP computational results for 20 instances, using the IP1, IP2 and LP2 (Halper et al., 2015).

Instance	IP1	IP1 RT(s)	IP2	IP2 RT(s)	IP-LP Gap (%)	LP2 RT(s)
2	4914	0.23	4914.00	0.15	0.00	0.08
3	5792	0.27	5792.00	0.13	0.00	0.06
4	4748	0.42	4748.00	0.13	0.00	0.06
5	2170	0.58	2170.00	0.11	0.00	0.04
8	6077	2.01	6077.00	1.65	0.00	1.06
9	4348	2.92	4348.00	0.91	0.00	0.33
10	2377	5.04	2377.00	0.68	0.00	0.20
13	5487	10.95	5487.00	8.09	0.01	5.30
14	3963	12.08	3963.00	2.76	0.00	2.23
15	2642	12.87	2642.00	1.46	0.00	0.53
18	5844	28.21	5844.00	14.48	0.00	10.11
19	4229	31.17	4229.00	8.00	0.00	4.01
20	3178	29.53	3178.00	6.83	0.00	1.76
23	6756	39.48	6756.00	8.13	0.00	5.30
24	4782	69.05	4782.00	5.96	0.00	2.06
25	3033	70.08	3033.00	6.37	0.00	1.90
28	6133	109.79	6133.00	130.34	0.02	5.32
29	4756	86.05	4756.00	20.94	0.02	2.95
30	3151	319.08	3151.00	8.70	0.00	2.98
33	6740	126.66	6740.00	17.21	0.00	10.67
Avg.		47.82		12.15	0.00	2.85
Max		319.08		130.34	0.02	10.67

Table 11: Computational results of the MFLP using 1-OptSwap and 1-SmartSwap (Halper et al., 2015).

Instance	1-OptSwapFI		1-OptSwapBI		1-SmartSwapFI		1-SmartSwapBI	
	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)	Gap (%)	Runtime(s)
2	0.39	0.02	0.00	0.01	0.00	0.00	0.00	0.02
3	0.00	0.02	0.00	0.03	1.28	0.00	0.00	0.00
4	0.00	0.03	0.31	0.02	1.41	0.02	1.30	0.02
5	0.58	0.09	0.58	0.31	4.47	0.00	0.58	0.02
8	0.75	0.06	0.03	0.26	0.76	0.02	0.03	0.08
9	0.38	0.37	0.24	2.42	2.10	0.02	0.63	0.19
10	0.37	1.44	0.22	8.01	1.49	0.05	1.60	0.22
13	0.01	0.48	0.01	2.20	0.53	0.08	1.22	0.28
14	0.00	2.20	0.63	16.18	1.35	0.13	1.06	0.92
15	0.27	5.62	0.00	56.46	1.44	0.13	0.00	1.06
18	0.15	0.70	0.05	7.60	1.69	0.13	0.10	1.20
19	0.25	4.81	0.36	67.39	0.48	0.47	1.00	3.06
20	0.31	17.86	0.84	188.67	3.37	0.50	0.97	3.23
23	0.11	2.09	0.09	27.85	0.42	0.39	0.67	4.07
24	0.07	12.81	0.28	164.41	2.23	0.66	0.76	6.88
25	0.59	47.58	0.38	508.58	2.12	0.81	0.80	9.75
28	0.10	3.32	0.02	54.46	0.48	0.76	0.39	7.32
29	0.50	34.80	0.17	395.04	1.47	1.73	1.02	15.09
30	0.35	103.41	0.23	1156.71	3.38	1.55	0.31	19.03
33	0.26	8.81	0.10	100.14	0.37	1.76	0.75	13.29
Avg.	0.27	12.33	0.23	137.84	1.54	0.46	0.66	4.29
Max	0.75	103.41	0.84	1156.71	4.47	1.76	1.60	19.03

Appendix C

In this appendix, a description is provided of the Java CPLEX source code, developed in this thesis. The Java code consists of two classes: the ModelMFLP and the HungarianAlgorithm.

The ModelMFLP is the main class, consisting of 33 different methods. One of the most important methods in this class is the method main, functioning as a console in which one can enter which methods the program has to execute for which instance(s). This method calls the methods runInstances if multiple instances have to be runned.

For every instance, the runningMethod is called by the method main. This method runs the readFile and creatingSets as data preparation. Finally, this method calls all solution methods for the MFLP, determined in the main. The methods MFLP_ModelIP1 and textscMFLP_ModelIP2 can be called to run the MFLP exactly. For calculating lower bounds for the MFLP, the methods MFLP_ModelLP1 and MFLP_ModelLP2 can be called.

The facility assignment and client assignment problem are implemented in respectively FA_subproblem and CA_subproblem by using CPLEX and in respectively hungarianMethod and CA_problem in a more efficient way, where the hungarianMethod makes use of the second class HungarianAlgorithm. Because the last two methods are the most efficient, all methods will use these two methods for solving the facility assignment and client assignment problem.

Finally, the methods n_OptSwap, n_SmartSwap and n_ForwardSwap are implemented for solving the MFLP. All three methods make use of a subordinate method for performing each iteration and an assignment method for calculating the optimal assignment between two sets of old and new facility destinations.