Solving the Facility Location Problem using Graph Theory and Shortest Path Algorithms

AUGUST DENES

PATRICK ZARABI
Solving the Facility Location Problem using Graph Theory and Shortest Path Algorithms

AUGUST DENES

PATRICK ZARABI

Degree Projects in Optimization and Systems Theory (30 ECTS credits)
Degree Programme in Industrial Engineering and Management (120 credits)
KTH Royal Institute of Technology year 2018
Supervisor at LKAB Kiruna: Monika Emanuelsson
Supervisor at KTH: Per Enqvist
Examiner at KTH: Per Enqvist
Abstract

This thesis in systems engineering and optimization theory aims to solve a facility location problem within the context of a confined space with path and proximity constraints. The thesis was commissioned by LKAB Kiruna, to help in their decision of where to construct a new facility on their industrial premises. The facility location problem was divided into a main problem of finding the best position of the facility, and a sub-problem of how to model distances and feasible areas within this particular context. The distance and feasibility modeling was solved by utilizing graph theory to construct a graph representation of a geographic area and then obtain the necessary distances using Dijkstra’s shortest path algorithm. The main problem was then solved using a mixed integer linear programming formulation which utilizes the distances obtained through the Dijkstra algorithm. The model is also extended to not only decide the placement of one facility but to accommodate the placement of two facilities. The extended model was solved in three ways, a heuristic algorithm, a mixed integer non linear formulation and a mixed integer linear formulation.

The results concluded that the implementation of the single facility model was able to obtain optimal solutions consistently. Regarding the extension, the mixed integer linear formulation was deemed to be the best model as it was computationally fast and consistently produced optimal solutions. Finally, several model improvements are identified to increase the applicability to different cases. These improvements could also allow the model to provide more strategical and managerial insights to the facility location decision process. Some future research into metaheuristics and machine learning are also suggested to further improve the usability of the models.
Sammanfattning


Resultaten visade att implementeringen av det ursprungliga lagerplaceringsproblemet konsekvent kunde beräkna optimala lösningar. Den utökade modellen löstes bäst av den linjära heltalsoptimeringsimplementeringen, då denna implementering konsekvent resulterade i bäst (lågast) värde i målfunktion samt löste problemet med låg beräkningstid. Slutligen identifierades flertalet potentiella modellförbättringar som skulle kunna implementeras för att ge modellen mer generaliserbarhet. Detta skulle även innebära att modellen själv kan utvärdera hur många lager som bör byggas givet en satt budget. Såleds kan modellen även erbjuda mer strategiska beslut om dessa förbättringar implementeras. Ytterligare forskning skulle även kunna göras inom metaheuristik och maskininlärning för att ytterligare förbättra distansmodelleringen.
Foreword
This thesis was commissioned by Luossavaara-Kiirunavaara Aktiebolag (LKAB) to help determine the placement of a potential new storage facility. The authors of the thesis have been working in collaboration with LKAB in order to determine the scope and desired results of the thesis.

The authors of this thesis would like to thank LKAB for providing an interesting problem which has enabled us to explore new ways of approaching and modeling mathematical problems. We are also thankful for the opportunity and the confidence placed in us throughout the process of writing this thesis. We would also like to thank our supervisor Per Enqvist, for continuously providing valuable insights and for pushing us to explore new avenues and aspects of the problem.

Sincerely,

August Dénes & Patrick Zarabi
# Contents

## 1 Introduction  
1.1 Background ................................................. 8  
1.2 Problem Formulation ......................................... 9  
1.3 Delimitation ................................................. 10  
1.4 Outline ................................................. 10

## 2 Theory  
2.1 Graph Theory ................................................. 11  
2.2 Algorithms ................................................. 12  
  2.2.1 Bellman-Ford Algorithm .................................. 12  
  2.2.2 Dijkstra’s Algorithm .................................... 13  
  2.2.3 Greedy Algorithm ..................................... 15  
2.3 Integer Programming ......................................... 15  
  2.3.1 Logical Constraint .................................... 16

## 3 Modeling  
3.1 Distance Modeling ........................................... 17  
3.2 Single Facility Location Problem ................................ 22  
3.3 Two Facility Joint Location Problem  
  3.3.1 Heuristic Algorithm ..................................... 23  
  3.3.2 Mixed Integer Non-Linear Programming .................. 25  
  3.3.3 Mixed Integer Linear Programming ..................... 27  
3.4 Implementation ............................................. 28  
  3.4.1 Distance Modeling .................................... 28  
  3.4.2 Single Facility Location Problem ....................... 29  
  3.4.3 Heuristic Algorithm .................................... 29  
  3.4.4 Two Facility MINLP and MILP  

## 4 Results  
4.1 Single Facility Location Problem  
  4.1.1 Test Cases ........................................... 30  
  4.1.2 LKAB ............................................. 33  
4.2 Two Facility Joint Location Problem  
  4.2.1 Test Cases ........................................... 34  
  4.2.2 LKAB ............................................. 38  
  4.2.3 Comparison ........................................... 41

## 5 Discussion  
5.1 Optimality ................................................ 44  
5.2 Distance Modeling and Implementation ..................... 45  
5.3 Two Facility Joint Location Problem  
5.4 Model Improvements and Further Research .................. 46

## 6 Conclusions  

List of Figures

1. Example of a graph, $G = (V,E)$, where $V = A, B, C, D$ and $E = (A, B), (A, C), (B, C), (B, D)$ .......................... 11
2. An undirected graph G with 4 vertices and 5 edges ...................... 12
3. Step by step, Bellman-Ford algorithm, at a) all the nodes are put to infinity before the relaxation is started in b-c) .......................... 14
4. LKAB’s industrial area in Kiruna .............................................. 17
5. Implementing a transparent grid over the map (only an example, not the grid used for the matrix in figure [6]) ......................... 19
6. The map from figure [4] with unfeasible areas and demand facilities highlighted. The red colored squares are unfeasible areas and yellow squares represent demand facilities .............................. 19
7. General structure of a distance matrix, $a_{ik}$ denotes the distance from feasible node i to demand facility k ......................... 20
8. An example of a small grid that becomes a graph in figure 9 where the red colored squares are unfeasible ..................................... 21
9. The small grid in figure 8 as a graph ........................................... 21
10. Matrix Example ................................................................. 28
11. Input matrix: "5x5 trivial" ..................................................... 30
12. Graph corresponding to Figure 11; Facility placed in Node 5, Objective value 144.349 .......................................................... 31
13. Input matrix: "5x5 non trivial" .................................................. 31
14. Graph corresponding to Figure 13; Facility placed in Node 13, Objective value 144.852 .......................................................... 31
15. Input matrix: "10x10 trivial" ..................................................... 32
16. Graph corresponding to Figure 15; Facility placed in Node 29, Objective value 144.852 .......................................................... 32
17. Input matrix: "10x10 non trivial" .................................................. 32
18. Graph corresponding to Figure 17; Facility placed in either Node 36, Node 46 or Node 56, Objective value 315.562 (Non unique solution) .................................................. 33
19. Input matrix: "LKAB" ................................................................. 33
20. Graph corresponding to Figure 19; Facility placed in Node 64, Objective value 4.976 .......................................................... 34
21. The red square with an X corresponds to the location of the facility .......................................................... 34
22. Input matrix: "5x5 trivial" ..................................................... 35
23. Graph corresponding to Figure 22; Facility 1 placed in Node 5, Facility 2 placed in Node 21, Objective value 10. The same solution was obtained by all implementations .......... 35
24. Input matrix: "5x5 non trivial" .................................................. 35
25. Graph corresponding to Figure 24; Heuristic results: Facility 1 placed in Node 13, Facility 2 placed in Node 6, Objective value 96.568 .......................................................... 36
26. Graph corresponding to Figure 24; MINLP and MILP results: Facility 1 placed in Node 13, Facility 2 placed in Node 6, Objective value 86.568 .......................................................... 36
27. Input matrix: "10x10 trivial" ..................................................... 36
28. Graph corresponding to Figure 27; Facility 1 placed in Node 30, Facility 2 placed in Node 38, Objective value 96.568. The same solution was obtained by all implementations .......... 37
29. Input matrix: "10x10 non trivial" .................................................. 37
Graph corresponding to Figure 29: Facility 1 placed in Node 6, Facility 2 placed in Node 76, Objective value 170. The same solution was obtained by all implementations.

Input matrix: "LKAB".

Graph corresponding to Figure 31: Heuristic solution: Facility 1 placed in Node 82, Facility 2 placed in Node 7, Objective value 1.357.

Graph corresponding to Figure 31: MILP solution: Facility 1 placed in Node 82, Facility 2 placed in Node 7, Objective value 1.287.

Graph corresponding to Figure 31: MINLP solution: Facility 1 placed in Node 99, Facility 2 placed in Node 7, Objective value 1.843.

The red squares with an X correspond to where the facilities are placed according to the MILP solution. Since this solution provides the best objective value, only this solution is illustrated geographically.
List of Tables

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Examples of Logical conditions</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Results from the trivial 5x5 matrix (Figure 22)</td>
<td>41</td>
</tr>
<tr>
<td>3</td>
<td>Results from the non trivial 5x5 matrix (Figure 24)</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td>Results from the trivial 10x10 matrix (Figure 27)</td>
<td>42</td>
</tr>
<tr>
<td>5</td>
<td>Results from the non trivial 10x10 matrix (Figure 29)</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Results from LKAB matrix (Figure 31)</td>
<td>43</td>
</tr>
</tbody>
</table>
1 Introduction

This section will provide necessary background information to illustrate the importance of the problem as well as a brief overview of previous literature on the subject. Furthermore, the problem formulation will be introduced and narrowed down to the research question of the thesis. Following the problem formulation, the thesis delimitations will be presented and to conclude the introduction, an outline of the report and its contents will be provided.

1.1 Background

LKAB is a government-owned, Swedish mining company. They are a worldwide, well-known iron ore provider. Their main products consist of fines and pellets, both of which are different forms of processed iron used in the production of steel. As a provider of iron, LKAB is acting in a commodity market. This means that the market dynamics are quite different from a typical goods market in terms of possible market strategies. Since commodities do not differ significantly in attributes, the possibility of using a differentiation strategy or offering additional value compared to competitors is limited. This notion has been explored by Sinclair S. & Seward K. in [1] which concluded that the market’s most valued attributes when choosing a supplier was the price and availability of the product. In other words, to gain a competitive advantage, LKAB needs to be able to offer their products at a competitive price as well as sustaining an efficient production. Therefore, cost minimization is of importance in all decisions and within all stages of production.

Given the competitive circumstances, the decision of where to place a new storage facility should be made with cost minimization in mind. Hence, the subject of the thesis is a single facility location problem. The facility location problem is well explored in scientific literature. One way to formulate the problem is the so-called Weber problem. The Weber problem utilizes the euclidean distance between the demand site and the facility to be placed [2]. Extensions to the Weber problem might also be made to accommodate vast distances, an example could be found in [3] where the great-circle distance is used to approximate railway distances. Another approach is presented in [4], where a conurbation is divided into twelve areas. The distances between each of these areas are then estimated from the center of the starting area to the center of the destination. In the specific case of LKAB, neither of these approaches are sufficiently applicable. The geographic area is too small to estimate distances using large-circle distance. Due to the size, the traveling distances also need to be well defined, eliminating the method used in [4] since there are no well suited partitions of LKAB’s industrial premises. This is a consequence of the fact that their production facilities are scattered across the premises along with several inaccessible areas. Furthermore, neither of the above methodologies explicitly account for path and location constraints, both of which need to be present in the case of LKAB due to the area containing several buildings and surfaces which are not possible to traverse. Additionally, the nature of the goods to be stored within the new facility requires safety constraints to account for proximity to other buildings.

Path constraints have been implemented by N. Katz & L. Cooper [2], where the forbidden region is defined to be a circle, and hence possible to express as a continuous and differentiable function. In the case of LKAB, the forbidden regions are irregular.
and scattered making them difficult to model as functions in the plane. The case illustrated in [2] also only had one forbidden region which was implemented. In order to account for all the obstructions present within LKAB’s premises, a vast amount of circles would have to be defined of varying sizes as to exclude all untraversable areas. The circles would also need to be further calibrated to ensure that no unintentional loss of accessibility is achieved.

As mentioned previously, there exists a lot of literature on the facility location problem. However, given the specific context of proximity and path constraints within a fairly small region, additional modeling is needed. Therefore, this thesis aims to solve the facility location problem within the context of a small geographic area with regards to both placement and path constraints.

1.2 Problem Formulation

The facility location problem is well explored in literature and different approaches, models and formulations have been constructed. Although some models have been developed to incorporate path restrictions or forbidden areas, the combination of path restrictions and placement/proximity constraints need additional modeling. Further, as many models aim to solve problems for vast distances, the euclidean distance and great-circle distance are common approximations. In this case, the area to be modeled is approximately 2.7x2.7 km$^2$. The use of bee line approximations are therefore not accurate. If e.g. an additional 100 m of traveling distance is added due to a building restricting the path, the bee line approximations are significant underestimations of the actual distance. Therefore, the distance modeling need to simulate the actual paths traveled.

The applicability of a model accounting for path and locations constraints within confined areas would definitely be useful to many actors considering a new facility. The biggest challenge in enforcing path and location constraint lies in the formulation of said constraints. The formulation also depends on how an area is defined, in the case of this thesis the starting point is a map of the area to be modeled. The initial problem to be solved is then how a map can be utilized to create a definition of the area. Once the area has been defined, the next step would be to determine how to model distances and paths. All of these components need to be incorporated into a model to be able to solve the facility location problem in the presence of path and location/proximity constraints within a confined area.

The thesis aims to provide a model capable of solving the facility location problem described. The research question is therefore formulated as: How to solve the facility location problem in the presence of path and location constraints?

In order to answer the research question, the following sub-problem needs to be solved:

- How to model distances and paths in a confined area with path and location constraints?
1.3 Delimitation

This thesis does not consider operational decisions of whether to build a facility or not. Instead it considers only the decision of where to build a facility given that a decision to build has been made. Furthermore, the thesis will not consider larger areas since the specified problem the thesis was commissioned to solve concerns a small area. Lastly, the thesis is based on the assumption that the objective of the model is to find the shortest possible distances between demands and facility location. Therefore, costs are assumed to be proportional to distance and no models will be considered with alternative objectives.

1.4 Outline

This thesis will implement a combination of graph theory, shortest path algorithms, optimization theory and heuristics to solve the facility location problem. The initial problem consist of a single facility, however, several model extensions will also be presented to solve a two facility location problem. Additional extensions to the model will also be discussed to further improve the applicability to a general problem.

The following chapters will introduce different algorithms, the modeling used to solve the facility location problem, the results obtained by the different models, a discussion and finally, a conclusion.


2 Theory

This section will provide a brief introduction to the theory, concepts and algorithms used throughout this thesis.

2.1 Graph Theory

In mathematics, a graph is a structure that connects some or all sets of objects. The objects correspond to mathematical abstractions called vertices (also called nodes or points) and each of the related pairs of vertices are called edges (also called arcs or lines). Graphs are useful for solving real-world problems and analyzing “things that are connected to other things”, particularly for solving different types of optimization problems. Hard problems can become easy when represented in a graph [5, p. 19].

Graphs are represented mathematically as \( G = (V, E) \) where \( V \) is a set of vertices and \( E \) is a set of ordered pairs of vertices, i.e. arrows, directed edges, directed lines or directed arcs see Figure 1.

![Figure 1: Example of a graph, \( G = (V, E) \), where \( V = A, B, C, D \) and \( E = (A, B), (A, C), (B, C), (B, D) \)](image)

The common practice to represent a graph is by using adjacency lists or adjacency matrices. Either method applies to both directed and undirected graphs. The directed graph is the graph represented in Figure 1 which has edges with direction. An example of an undirected graph is shown in Figure 2 which does not have edges with direction.

The preferable method of representing sparse graphs is with adjacency lists, since \(| E |\) is much less compared to \(| V |^2\) in space. However, when the graph is dense - \(| E |\) is close to \(| V |^2\) then an adjacency matrix would be preferable [6, p. 589].
2.2 Algorithms

The definition of an algorithm is: A gradual procedure of well-defined executable instructions, defined to perform a task or to solve a problem often with the requirement that the procedure has an end.

The word "Algorithm" originates from the Persian mathematician, Abu Ja’far Mohammed ibnMusa al-Khowarizm who lived in the ninth century.

An algorithm can be seen as a mathematical recipe, consisting of a set of rules to be performed systematically i.e. an outcome to the solution of a well-defined problem. The algorithm can be seen as strictly deterministic or stochastic in nature. Additionally, the algorithms can either be sequential or parallel, where in the sequential algorithm the steps that are forming the algorithm are ordered and should be performed one after the other. In parallel algorithms there exist rules that are to be enforced simultaneously [7, p. 529-533].

2.2.1 Bellman-Ford Algorithm

The Bellman-Ford algorithm is a single-source-shortest-path finding algorithm, which calculates the shortest paths from a single source node to all the other nodes in a weighted and directed graph \( G = (V,E) \). In contrast to Dijkstra’s algorithm in subsection 2.2.2, the Bellman-Ford algorithm allows for edge weights to be negative. The algorithm initiates by marking a parent node (start node) \( s \) with a distance \( d_0 = 0 \). To every other node \( u \), a distance value of \( d_u = \infty \) is assigned. The values that are given to every \( u \) node, are temporary values. Subsequently, the algorithm explores all the edges trying to see if there is a possibility to relax them. This method is called the relax method and is also used in Dijkstra’s algorithm. The relaxation is operating \( |V| - 1 \) times, where \( V \) is the number of vertices in the graph. If the exploration gets finished successfully, the graph has no negative cycles and the data that you compute is correct, the algorithm stops [8].

Figure 2: An undirected graph \( G \) with 4 vertices and 5 edges.
Algorithm 1 Bellman-Ford [6] p.651

\[
\text{Initialize Single Source } (G, s) \\
\text{for } i=1 \text{ to } |G.V| - 1 \text{ do} \\
\quad \text{for each edge } (u, v) \in G.E \text{ do} \\
\quad \quad \text{RELAX}(u, v, w) \\
\quad \text{end for} \\
\text{end for} \\
\text{for each edge } (u, v) \in G.E \text{ do} \\
\quad \text{if } v.d > u.d + w(u, v) \text{ (where } w \text{ is the weight) then} \\
\quad \quad \text{return False} \\
\quad \text{end if} \\
\text{end for}
\]

As described above, the operation is performed \(|V| - 1\) times. Hence, the time complexity of the algorithm is \(O(VE)\) since the initialization takes \(\Theta(V)\) time and each of the \(|V| - 1\) passes over the edges takes \(\Theta(E)\) time. In other words, Bellman-Ford performs \(|E|\) relaxations for every iteration and there are \(|V| - 1\) iterations. Additionally, the space complexity in the worst case scenario is \(O(V)\).

Advantages

1. It is a dynamic Algorithm.
2. It works with negative edge weights.
3. It only requires local information. Hence, easy to implement. [9]

Disadvantages

1. It only works for directed graphs. [9]

2.2.2 Dijkstra’s Algorithm

Dijkstra’s algorithm is an algorithm for solving the single-source shortest-paths (SSSP) problem in a weighted and directed graph \(G = (V, E)\) where all edge weights are non-negative. The algorithm maintains a set \(S\) of nodes whose final shortest-path weights from the source \(s\) have already been determined.

The objective of the algorithm is to find the lowest cost, i.e., the shortest path from the start node \(s\) to every other node. It repeatedly selects the node \(u \in V - S\) with the minimum shortest-path estimate, adds \(u\) to \(S\), and relaxes all edges leaving \(u\). In the following implementation, a min-priority queue \(Q\) of nodes is used. The nodes are keyed by their \(d\) values [6] p. 658].

The time complexity of the Dijkstra algorithm depends on a combination of the underlying data structure and the graph shape. The original Dijkstra algorithm does not use a min-priority queue, which means that the time complexity is \(O(|V|^2)\). However, an
implementation of the min-priority (min-heap) queue will reduce the running time to $O(|E| + |V|\log |V|)$. This is the fastest known SSSP algorithm for directed graphs with non-negative weights \([10]\).

As represented in the algorithm\(^2\) Dijkstra’s algorithm is working backwards from the end to the beginning, trying to find the shortest leg each time.

**Algorithm 2 Dijkstra\([6, p.658]\)**

1. Initialize Single Source \((G, s)\)
2. \(S = \emptyset\)
3. \(Q = G.V\)
4. while \(Q \neq \emptyset\) do
5.   \(u = \text{Extract Min}(Q)\)
6.   \(S = S \cup (u)\)
7.   for each edge \(v \in G.\text{Adj}(u)\) do
8.     RELAX\((u, v, w)\)
9.   end for
10. end while

**Advantages**

1. It is a Greedy Algorithm. Which means it is easy to implement.

2. It works for directed and undirected graphs.\([9]\)

**Disadvantages**

1. It does not work with negative edge weights.
2. It requires global information. Hence, higher time complexity.

2.2.3 Greedy Algorithm

A simple way of solving optimization problems is by using a greedy algorithm. It is an algorithm that makes the choice that looks best at the moment. This means that it basically makes the locally optimal choice to try to find a globally optimal solution. Hence, the greedy algorithm does not always yield an optimal solution since it could stop in a local optimal solution.

The greedy algorithm is a powerful method to solve different types of optimization problems. Classic examples of algorithms that are using the application of greedy methods are: The minimum-spanning tree algorithm, Dijkstra’s algorithm for shortest paths from a single source and Chvatal’s greedy set-covering heuristic [6, p.414].

An example of how the greedy algorithm operates; a busy businessman have exactly $T$ hours to perform some tasks and he wants to maximize the amount of completed tasks within those $T$ hours. An array $A$ of integers are given, where each element indicates the time it takes to complete a task.

In each iteration, the businessman has to greedily select the tasks which will take the minimum amount of time to complete. This example is a simple greedy-algorithm problem [11].

Advantages

1. It is quite easy to come up with a greedy algorithm for an optimization problem.
2. Analyzing the run time for greedy algorithms will generally be much easier than other techniques.

Disadvantages

1. It is difficult to prove the correctness of the the greedy algorithm, in general it is hard to prove why it is correct.

2.3 Integer Programming

Integer programming (IP) also called Integer Linear programming (ILP) is a mathematical technique to find the best solution in allocating limited resources for an optimization problem. In fact, that also applies for Linear programming (LP), although, in many practical problems it is common that the decision variables are integer valued.

The mathematical model for IP is the LP model but with the restriction that the variables are required to be integer valued. In some cases there are problems that only requires some of the variables to be integer valued, these models are referred to as
mixed integer programming (MIP) \[12\] p. 474].

The importance of the application of IP is seen in problems involving a number of interrelated “yes-or-no decisions”. In such situations there are only two possible choices, either yes or no. Example of these problems are: Should we make an investment? Should we allocate a facility to a particular site? Usually, the representation of these two choices can be made as binary variables, see Equation 1.

\[
x_j = \begin{cases} 
1 & \text{if decision } j \text{ is yes} \\
0 & \text{if decision } j \text{ is no} 
\end{cases} \quad (1)
\]

2.3.1 Logical Constraint

In the previous subsection 2.3, the use of binary variables were introduced which represent yes and no decisions. Furthermore, binary variables can also be used in connection with relationships called logical constraints. Logical constraints restrict consideration to certain combinations of variables \[13\] p. 251-296]. An example of this would be if a project manager had to choose between two projects to invest his time in. He could only choose one of the projects. The projects are represented as \(x_1\) and \(x_2\), and are binary variables. Since the project manager could just do one of the projects, there is an equation represented for that in Equation 2. This constraint forces the project manager to only choose one project. There are several logical constraints that can be used in LP and IP problems. In Table 1, some of these conditions for logical constraints are displayed.

\[
x_a + x_b \leq 1 \quad (2)
\]

<table>
<thead>
<tr>
<th>Condition</th>
<th>Logical constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>(x_A + x_B = 2)</td>
<td>A and B</td>
</tr>
<tr>
<td>Or</td>
<td>(x_A + x_B \geq 1)</td>
<td>A or B, or both</td>
</tr>
<tr>
<td>Exclusive or</td>
<td>(x_A + x_B = 1)</td>
<td>A or B but not both</td>
</tr>
<tr>
<td>If-then</td>
<td>(x_A \leq x_B)</td>
<td>B exists if A exists</td>
</tr>
<tr>
<td>If and only if</td>
<td>(x_A = x_B)</td>
<td>A if and only if B</td>
</tr>
</tbody>
</table>

Table 1: Examples of Logical conditions
3  Modeling

This section will present the modeling of the mathematical formulations to the research questions posed in subsection 1.2. Firstly, the proposed solution to the distance and feasible area modeling will be explained. Secondly, the single facility location problem will be presented in terms of mathematical formulation and implementation. Lastly, an extension of the single facility problem will be formulated in order to allocate two facilities jointly. The extended problem will be implemented using three different models.

Figure 4: LKAB’s industrial area in Kiruna

3.1  Distance Modeling

The initial problem of formulating a facility location model is how to define the distance between an origin and the demand facilities to be served. As mentioned in the introduction, the geographical area to be modeled is fairly small. This means that, if e.g. a euclidean distance formulation were to be used, the distances could be significantly underestimated. The euclidean distance would be the shortest possible straight line from the origin to the destination, i.e. the bee line. However, if e.g. the line were to cross through a building, the actual distance that would have to be traveled around the building would not be represented in the model.

The specifics of LKAB’s problem also require additional constraints regarding proximity to other buildings. This means that the chosen location for the facility can not be in direct contact to any other buildings. Furthermore, the area contains several inacces-
sible regions, meaning that neither traveling through these regions or the placement of a facility within these regions is feasible. These conditions make it difficult to model the distances by conventional distance measures, e.g. euclidean distance or great-circle distance.

In order to account for all the above constraints to the area, a similar approach as shown in [3] is implemented. Namely, to introduce a uniform grid across the entire area. Each square within this grid is represented by it’s centroid, i.e. the distance from one square to the next is defined as the distance between their centroids. This is where the similarities to [3] end as the distances in [3] are then defined by the great-circle distance.

This approach provides flexibility, as the mathematical model is not inherently interested in the actual distances traveled, but rather in finding the shortest paths possible. By defining the distance between two neighbouring squares as one unit, the actual distance can be calculated by taking the amount of units traveled multiplied by the distance the breadth of one square represents within the area. The objective of the model will therefore be to minimize the amount of units traveled. This approach also implies that the accuracy can be adjusted, by increasing the amount of squares within the applied grid (decreasing the size of each square) the accuracy can be improved. However, as mentioned previously, the proximity constraints inherent to LKAB’s problem prevents us from making the squares too small. By having the squares slightly larger, the proximity constraints can still be fulfilled even if the square chosen contains a small building or a part of a building. This also means that the size of the squares can be adjusted to accommodate larger or smaller facilities. Furthermore, if the square chosen is empty, a slightly larger area would still be preferable. This would then account for other factors that might not allow the facility to be placed at an exact spot, e.g. building permits. However, these are problems specific to LKAB. The benefit of this approach is that the squares can be adjusted to fit any particular problem. In other words, the mathematical model does not depend on the size of the squares and hence, for any other application the size could be altered without any alterations to the model. The area modeled is shown in [Figure 4] the grid application is shown in [Figure 5]
Figure 5: Implementing a transparent grid over the map (only an example, not the grid used for the matrix in figure 6).

Figure 6: The map from figure 4 with unfeasible areas and demand facilities highlighted. The red colored squares are unfeasible areas and yellow squares represent demand facilities.
Once the grid is applied to the area being modeled, each centroid is translated into a node in an undirected graph. In other words, each square in the grid will correspond to a node in a graph structure. Each node will then be connected to its neighbouring nodes by an undirected arc, given that the neighbouring node is feasible in terms of facility placement and accessibility. This means that if a square contains a large building which occupies most of the space within the square, or if the square contains an inaccessible area, that square is deemed unfeasible. This means that the node in the graph corresponding to this unfeasible square in the grid, will not be connected to its neighbours. Thereby, this node can not be traversed through, simulating that the building or inaccessible area is obstructing the path. Similarly, due to the node lacking any connections, there are no paths available from this node to any demand sites. Thereby, the node is also unfeasible in terms of facility placement. Since the unfeasible nodes do not possess any connections, they are illustrated as gaps in the graphs.

The structure of the graph will mimic the grid structure applied to the area. Thereby, each node can have at most eight arcs connecting it to neighbouring nodes, two connections vertically, two connections horizontally and four connections diagonally. By utilizing a graph structure, the shortest path between any two feasible nodes can be obtained by graph based search algorithms or shortest path algorithms. In this thesis, the Dijkstra shortest path algorithm is used to obtain the shortest paths from each feasible node to each node containing a demand facility. These distances can then be exported in matrix form to be used in the mathematical models. The structure of the distance matrix obtained is as follows:

- Each row corresponds to an origin node
- Each column corresponds to a node containing a demand facility.

See the following example:

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & \cdots & a_{1k} \\
  a_{21} & a_{22} & a_{23} & \cdots & a_{2k} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  a_{i1} & a_{i2} & a_{i3} & \cdots & a_{ik}
\end{bmatrix}
\]

Figure 7: General structure of a distance matrix, \(a_{ik}\) denotes the distance from feasible node \(i\) to demand facility \(k\).
Figure 8: An example of a small grid that becomes a graph in figure 9 where the red colored squares are unfeasible.

The benefit of using the grid to create a graph is that the distances obtained with Dijkstra will be a representation of the actual path traveled. This means that any obstruction to the bee line path, e.g. a building, will render that path unfeasible. The algorithm will
therefore not be able to choose that path. Hence, the algorithm will be forced to choose a route around the obstruction. By utilizing this method, the distances obtained will closer mimic the actual path and distances in comparison to the euclidean distance modeling. However, it is worth noting that the distances obtained will still be an approximation. The accuracy of the distances will be dependent on the size of the squares in the grid application. Smaller squares will provide a more accurate path at the cost of increased amounts of nodes and thereby increase the complexity of the problem.

3.2 Single Facility Location Problem

The single facility location problem can be quite easily formulated given a distance matrix obtained by Dijkstra as described in the previous section. The single facility problem is formulated as a mixed integer linear programming problem (MILP) as shown in Equation 3.

\[
\text{Minimize: } \sum_{i \in I} \sum_{k \in K} x_i w_k a_{ik} \tag{3a}
\]

Subject to:

\[
\sum_{i \in I} x_i = 1 \tag{3b}
\]

\[
x_i \in \{0, 1\} \tag{3c}
\]

The above formulation minimizes the weighted sum of distances from node \(i\) to each demand facility \(k\). The binary variable \(x_i\) indicates which node the facility should be placed in. The constraint \(3b\) ensures that the algorithm assigns a facility to one of the nodes. Without the constraint, the objective value would be zero since the most optimal choice would be to not allocate any new facilities given the formulation in \(3a\). Therefore, the formulation above merely needs to choose the best location possible and not evaluate feasibility.

3.3 Two Facility Joint Location Problem

This section will provide model formulations for an extension to the single facility location problem. Namely, a two facility joint location problem. The aim of the models
is to allocate two facilities who each will serve a subset of the demands in $K$. The premise of the problem is that no input regarding partitioning of the demands should be given. The models should optimally divide the demands between the two facilities. Furthermore, there are no constraints regarding the amount of demands served by either facility, i.e. one of the facilities could end up serving just one demand while the second facility serves all of the remaining demands. A key aspect to note is that neither of the models aim to decide whether one or two facilities is most beneficial. The models simply determine the two best locations to place the facilities and the optimal partitioning of the demands.

The joint facility location problem is solved by three different models. The models are formulated sequentially and are presented in the order in which they were conceived. The first model to be shown is a heuristic algorithm. As heuristic algorithms can be difficult to evaluate in terms of optimality, a comparison mixed integer non-linear programming (MINLP) model is formulated. Lastly, in an attempt to simplify the MINLP model and improve computation time, a mixed integer linear programming model is formulated. All the models are described in detail in the following subsubsections.

### 3.3.1 Heuristic Algorithm

The heuristic approach to solving the problem of assigning two facilities to serve a given number of demands utilizes the same distance modeling as shown previously. The distances $a_{ik}$ are therefore obtained by Dijkstra’s algorithm. The heuristic is based on a greedy set partitioning method. This means that the algorithm divides the demands to be served into two sets. One set to be served by facility 1 ($S_1$) and the other to be served by facility 2 ($S_2$). The algorithm then calculates the total weighted sum of distances traveled and reassigns the demand sets ($S_1$ and $S_2$) to decrease the total weighted sum of distances. The algorithm reassigns one demand in every iteration and compares the new weighted sum to the previous iteration. Once the total weighted sum is no longer improved, the algorithm stops and the solution is returned. The notation and algorithm is described in detail below.

<table>
<thead>
<tr>
<th>Set</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$i$</td>
<td>All feasible nodes $i$ in the graph</td>
</tr>
<tr>
<td>$K$</td>
<td>$k$</td>
<td>Nodes $k$ containing a demand facility, $K \subseteq I$</td>
</tr>
<tr>
<td>$S_1$</td>
<td></td>
<td>Set of demands served by facility 1, $S_1 \subseteq K$, $S_1 \cup S_2 = K$</td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td>Set of demands served by facility 2, $S_2 \subseteq K$, $S_1 \cup S_2 = K$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Param</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_k$</td>
<td>$\mathbb{R}$</td>
<td>Weight associated with demand $k$</td>
</tr>
<tr>
<td>$a_{ik}$</td>
<td>$\mathbb{R}$</td>
<td>Distance from node $i$ to demand $k$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Integer</td>
<td>Location of facility 1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Integer</td>
<td>Location of facility 2</td>
</tr>
<tr>
<td>$y_{1,i}$</td>
<td>Binary</td>
<td>Facility 1 placed in node $i$</td>
</tr>
<tr>
<td>$y_{2,i}$</td>
<td>Binary</td>
<td>Facility 2 placed in node $i$</td>
</tr>
<tr>
<td>$Z$</td>
<td>Float</td>
<td>Total weighted sum of distances</td>
</tr>
</tbody>
</table>
\begin{align*}
  x_1 &= \begin{cases} 
    \text{Minimize:} & \sum_{i \in I} \sum_{k \in S_1} y_{1,i} w_k a_{ik} \\
    \text{Subject to:} & \sum_{i \in I} y_{1,i} = 1 \\
    & y_{1,i} \in \{0, 1\} 
  \end{cases} \\
  x_2 &= \begin{cases} 
    \text{Minimize:} & \sum_{i \in I} \sum_{k \in S_2} y_{2,i} w_k a_{ik} \\
    \text{Subject to:} & \sum_{i \in I} y_{2,i} = 1 \\
    & y_{2,i} \in \{0, 1\} 
  \end{cases} \\
  Z &= \sum_{k \in S_1} a_{x_1,k} w_k + \sum_{k \in S_2} a_{x_2,k} w_k 
\end{align*}

**Algorithm 3** Two Facility Heuristic

1. **Initialize two facility heuristic**
   - $S_1 = K$ (Assign all demands to be served by facility 1)
   - $S_2 = \emptyset$ (Facility 2 has not yet been assigned any demands)
2. Obtain $x_1$ by solving **Equation 4**
   - $x_2 = \text{None}$ (Facility 2 will not yet be placed)
3. **Total Cost** = $Z$ (Calculate $Z$ from **Equation 6**)
4. **Updated cost** = **Total cost** (Set an initial value to **Updated cost**)
5. **while** **Updated cost** $\leq$ **Total Cost** **do**
   - **Total cost** = **Updated cost** (Set the current cost $Z$ as new benchmark value)
   - $S_{2+} = \max_k (a_{x_1,k} w_k)$ (Reassign the demand $k$ with the largest weighted distance from $x_1$ to be served by facility 2)
   - $S_{1-} = \max_k (a_{x_1,k} w_k)$ (Remove the demand $k$ with the largest weighted distance from $x_1$ from $S_1$)
   - Obtain the new position $x_1$ by solving **Equation 4** with the updated set $S_1$
   - Obtain the position $x_2$ by solving **Equation 5** with the updated set $S_2$
   - Calculate the new total cost $Z$ from **Equation 6**
   - **Updated cost** = $Z$ (Assign the updated cost to be the updated value of $Z$)
6. **end while**
7. return **Total Cost**

The algorithm starts by assigning all demands in $K$ to be served by facility 1. The algorithm then solves the single facility location problem to determine the location $x_1$. Once the location has been determined for the single facility, the total weighted sum of distances is calculated and set as the initial benchmark. The algorithm then determines which demand contributes the most to the objective value, i.e. which demand has the largest weighted distance. The demand with the largest weighted distance from $x_1$ is then assigned to $S_2$ to be served by facility 2 and thereby the demand is removed from $S_1$. New positions for $x_1$ and $x_2$ can now be determined given the alterations to $S_1$ and $S_2$. The new total weighted sum is calculated and compared to the initial benchmark. If the updated value is less or equal to the benchmark, the benchmark is updated. The algorithm repeats the process of determining which demand in $S_1$ has the largest weighted distance from $x_1$ and reassigns this demand to $S_2$. New positions $x_1$ and $x_2$ are calculated and a new total weighted sum $Z$ is determined. If the new updated value of $Z$ is less or equal to the current benchmark, the algorithm continues reassigning demands from $S_1$ to $S_2$ until no further improvement is obtained. When the updated
value no longer is smaller than the current benchmark, the algorithm terminates and returns the current positions \( x_1 \) and \( x_2 \) as well as the current lowest objective value.

The described heuristic has the advantage of being computationally fast since the algorithm is solving two separate single facility problems. Further, the choice of demand to reassign from \( S_1 \) to \( S_2 \) is also a simple calculation. Especially since the amount of demands will generally be quite small in comparison to the amount of feasible locations. The computational speed comes at the cost of not always being able to locate an optimal solution. In some cases the initial solution to the single facility problem (line 3 in 3) might not be unique. The algorithm will then pick the first solution and continue from there. The same scenario could also arise as the algorithm determines which demand to reassign to \( S_2 \). If there are several demands with the same weighted distance from \( x_1 \), the algorithm chooses the first of these. Hence, there are several feasible improvements the algorithm will not evaluate. Therefore, suggested solutions from the heuristic are in some cases sub-optimal. This will be illustrated in section 4.

In order to evaluate the optimality of the heuristic, a non-linear mixed integer program is developed, see subsubsection 3.3.2.

### 3.3.2 Mixed Integer Non-Linear Programming

The heuristic algorithm presented in the previous section has some inherent flaws due to the implementation. In order to evaluate the efficiency and accuracy of the heuristic, the following mixed integer non-linear programming formulation (MINLP) was constructed. As previously, the distances for the model are derived from Dijkstra’s algorithm. Furthermore, the MINLP formulation abides by the same assumptions as the heuristic. Namely, the model divides the amount of demands between the two facilities and no constraints are set regarding the proportion of demands served by either facility. The MINLP model does not decide whether or not a second facility should be built, it only determines the two best positions to place the facilities.

<table>
<thead>
<tr>
<th>Set</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>( i )</td>
<td>Set of feasible nodes ( i ) in graph associated with facility 1</td>
</tr>
<tr>
<td>J</td>
<td>( j )</td>
<td>Alias set to ( I ), feasible nodes in graph associated with facility 2, ( I = J )</td>
</tr>
<tr>
<td>K</td>
<td>( k )</td>
<td>Nodes ( k ) containing a demand facility, ( K \subset I )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Param</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_k )</td>
<td>( \mathbb{R} )</td>
<td>Weight associated with demand ( k )</td>
</tr>
<tr>
<td>( a_{ik} )</td>
<td>( \mathbb{R} )</td>
<td>Distance from node ( i ) to demand ( k ) for facility 1</td>
</tr>
<tr>
<td>( b_{jk} )</td>
<td>( \mathbb{R} )</td>
<td>Distance from node ( j ) to demand ( k ) for facility 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{i,1} )</td>
<td>Binary</td>
<td>Facility 1 placed in node ( i )</td>
</tr>
<tr>
<td>( d_{i,k} )</td>
<td>Binary</td>
<td>Facility 1 placed in node ( i ) serves demand ( k )</td>
</tr>
<tr>
<td>( x_{j,2} )</td>
<td>Binary</td>
<td>Facility 2 placed in node ( j )</td>
</tr>
<tr>
<td>( d_{j,k} )</td>
<td>Binary</td>
<td>Facility 2 placed in node ( j ) serves demand ( k )</td>
</tr>
<tr>
<td>( y_i )</td>
<td>Binary</td>
<td>An added variable used for an if-then constraint</td>
</tr>
</tbody>
</table>

\[
d_{i,k} = \begin{cases} 
1 & \text{when facility 1 serves demand } k \text{ from node } i \\
0 & \text{else} 
\end{cases} \quad (7)
\]
\[ d_{jk,2} = \begin{cases} 1 & \text{when facility 2 serves demand } k \text{ from node } j \\ 0 & \text{else} \end{cases} \quad (8) \]

Minimize:
\[ \sum_{i \in I} \sum_{k \in K} a_{ik} w_k x_{i,1} d_{ik,1} + \sum_{j \in J} \sum_{k \in K} b_{jk} w_k x_{j,2} d_{jk,2} \quad (9a) \]

Subject to:
\[ \sum_{i \in I} x_{i,1} = 1 \quad (9b) \]
\[ \sum_{j \in J} x_{j,2} = 1 \quad (9c) \]
\[ x_{i,1} \leq y_i \quad \forall i \quad (9d) \]
\[ x_{j,2} \leq 1 - y_i \quad \forall i = j \quad (9e) \]
\[ d_{ik,1} \leq x_{i,1} \quad \forall i, k \quad (9f) \]
\[ d_{jk,2} \leq x_{j,2} \quad \forall j, k \quad (9g) \]
\[ \sum_{i \in I} d_{ik,1} + \sum_{j \in J} d_{jk,2} = 1 \quad \forall k \quad (9h) \]

Compared to the single facility location problem, the two facility joint location problem formulation requires the introduction of several additional variables. The first variable to be added is the location of the second facility \( x_{j,2} \). The second and third additional variables, \( d_{ik,1} \) and \( d_{jk,2} \) are meant to act as demand assignment variables. In other words, which demand should be serviced by which facility. These variables are necessary in order for the model to partition the demands amongst the two facilities without any prespecified assignments.

The objective function [Equation 9a] is in essence identical to the single facility objective in [Equation 3]. They both are meant to minimize the weighted sum of traveled distances. The difference is as previously stated, the introduction of the binary demand assignment variables. The demand assignment variables make the objective function non-linear due to the multiplication of the two binary variables \( x, d \). Furthermore, the constraints [Equation 9b] and [Equation 9c] provide the same functionality as [Equation 3b] that is to force the placement of facility 1 and facility 2 and limit the amount of facilities placed to one. Constraint [Equation 9d] and [Equation 9e] ensure that facility 1 and facility 2 can not be placed in the same node. The constraints [Equation 9f] and [Equation 9g] are there to connect the binary variables \( x, d \) such that a demand \( k \) can be served by a facility from node \( i \) or \( j \) if and only if a facility is located in node \( i \) or \( j \). Lastly, constraint [Equation 9h] ensures that each demand is serviced once by one of the facilities.

The MINLP formulation above is difficult to solve as the size of the problem increases. The model is able to handle smaller problems during testing. However, as the amount of feasible locations increases, the computation time and resources needed increase significantly. This can be observed in section 4. This is due to MINLP problems being NP-hard [14]. Therefore, the following subsubsection 3.3.3 provides a linearization of the above MINLP formulation.
3.3.3 Mixed Integer Linear Programming

This section will present a linearization of the above MINLP model, i.e., a mixed integer linear programming (MILP) formulation.

<table>
<thead>
<tr>
<th>Set</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>$i$</td>
<td>Set of feasible nodes $i$ in graph associated with facility 1</td>
</tr>
<tr>
<td>J</td>
<td>$j$</td>
<td>Alias set to $I$, feasible nodes in graph associated with facility 2, $I = J$</td>
</tr>
<tr>
<td>K</td>
<td>$k$</td>
<td>Nodes $k$ containing a demand facility, $K \subset I$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Param</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_k$</td>
<td>$\mathbb{R}$</td>
<td>Weight associated with demand $k$</td>
</tr>
<tr>
<td>$a_{ik}$</td>
<td>$\mathbb{R}$</td>
<td>Distance from node $i$ to demand $k$ for facility 1</td>
</tr>
<tr>
<td>$b_{jk}$</td>
<td>$\mathbb{R}$</td>
<td>Distance from node $j$ to demand $k$ for facility 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{i,1}$</td>
<td>Binary</td>
<td>Facility 1 placed in node $i$</td>
</tr>
<tr>
<td>$d_{ik,1}$</td>
<td>Binary</td>
<td>Facility 1 placed in node $i$ serves demand $k$</td>
</tr>
<tr>
<td>$x_{j,2}$</td>
<td>Binary</td>
<td>Facility 2 placed in node $j$</td>
</tr>
<tr>
<td>$d_{jk,2}$</td>
<td>Binary</td>
<td>Facility 2 placed in node $j$ serves demand $k$</td>
</tr>
<tr>
<td>$y_i$</td>
<td>Binary</td>
<td>An added variable used for an if-then constraint</td>
</tr>
<tr>
<td>$s_{ik}$</td>
<td>Binary</td>
<td>Facility 1 is placed in node $i$ to serve demand $k$</td>
</tr>
<tr>
<td>$g_{jk}$</td>
<td>Binary</td>
<td>Facility 2 is placed in node $j$ to serve demand $k$</td>
</tr>
</tbody>
</table>

$s_{ik} = \begin{cases} 1 & \text{when facility 1 serves demand } k \text{ and is placed in node } i \\ 0 & \text{else} \end{cases}$ (10)

$g_{jk} = \begin{cases} 1 & \text{when facility 2 serves demand } k \text{ and is placed in node } j \\ 0 & \text{else} \end{cases}$ (11)

Minimize:

$$\sum_{i \in I} \sum_{k \in K} a_{ik} w_k s_{ik} + \sum_{j \in J} \sum_{k \in K} b_{jk} w_k g_{jk}$$ (12a)

Subject to:

$$\sum_{i \in I} x_{i,1} = 1$$ (12b)

$$\sum_{j \in J} x_{j,2} = 1$$ (12c)

$$x_{i,1} \leq y_i \quad \forall i$$ (12d)

$$x_{j,2} \leq 1 - y_i \quad \forall i = j$$ (12e)

$$d_{ik,1} \leq x_{i,1} \quad \forall i, k$$ (12f)

$$d_{jk,2} \leq x_{j,2} \quad \forall j, k$$ (12g)

$$\sum_{i \in I} d_{ik,1} + \sum_{j \in J} d_{jk,2} = 1 \quad \forall k$$ (12h)

$$s_{ik} \geq d_{ik,1} + x_{i,1} - 1 \quad \forall i, k$$ (12i)

$$g_{jk} \geq d_{jk,2} + x_{j,1} - 1 \quad \forall j, k$$ (12j)
The MILP formulation is very similar to the MINLP model. The difference in the objective function is the introduction of the binary variables $s$ and $g$. These new variables serve as substitutes for the multiplications of $x$ and $d$ in the MINLP model. Due to the similarities of the models, most of the constraints in the MILP model above are identical to the MINLP. The differences being the added constraints [Equation 12i] and [Equation 12j]. These constraints assign values to $s$ and $g$ based on the placement of the facilities (the $x$ variables) and the demand assignment (the $d$ variables). The constraints simply state that the value of $s$ and $g$ is equal to one, if the corresponding facility is placed in node $i$ or $j$ and the demand $k$ is assigned to that facility.

This model is significantly faster computationally in comparison to the MINLP model. This is due to the linear objective function. The differences are shown in section 4.

### 3.4 Implementation

This section describes the implementation of the previously presented model formulations. The order of which the implementations are described is the same as the order in which the model formulations are presented.

#### 3.4.1 Distance Modeling

The distance modeling implementation is made in Python 3. The implementation is based on the library Dijkstar available at the official python package site Python Package Index. The package includes a graph class as well as an implementation of Dijkstra’s algorithm. Using the existing graph class, a function for reading binary representations of an area in the form of a matrix was created. The matrix represents the grid application over the map, where each element in the matrix would correspond to a square in the grid. Each element in the matrix is then translated into a node in a graph. The matrix elements can consist of "1", indicating that the square is accessible and therefore a feasible facility location, "0", indicating that the square can not be traversed through and therefore an unfeasible location, and finally "w (float)", indicating that the square contains a demand facility with weight (float). See the following example.

\[
\begin{bmatrix}
1 & 1 & 0 \\
5 & 1 & 0 \\
0 & 1 & w 3
\end{bmatrix}
\]

Figure 10: Matrix Example

Utilizing the numbering of nodes as shown in Figure 8, this matrix should be interpreted as node 3, node 6 and node 7 being unfeasible. The nodes 4 and 9 contain a demand facility with weight 5 and 3 respectively. It is important to note that an underlying assumption in the creating of the graph is that a node containing a demand is seen as feasible. This implies that the solution to the placement problem could be a node containing a demand facility.
The function reads matrices on the form shown in Figure 10 with no restrictions to size. The only restriction is that each row needs to be of the same length. An undirected graph is then created in accordance with the matrix, where each node represents the center of the corresponding square in the applied grid. This means that there are no restrictions on the direction of traversing between two feasible nodes. The graph contains connections between all neighbouring feasible nodes, i.e., traversing vertically, horizontally and diagonally is possible. However, the amount of units travelled or cost of travel differs between vertical/horizontal traversing and diagonal traversing. The cost associated with traversing vertically or horizontally is by construction set to 1 unit as the length traveled corresponds to the width or height of one square in the applied grid. The diagonal cost is set to $\sqrt{2} \approx 1.4142$ obtained by Pythagora’s theorem. Lastly, the cost of traversing within a node, e.g., node 1 to node 1 is set to zero. Therefore, if a problem containing only one demand facility should be evaluated using the presented models, the objective value would be equal to zero due to the location being set to the same node as the demand facility is located in.

When the graph is created, the distance matrix containing all the shortest paths between feasible nodes and demand facilities is obtained by using the Dijkstra implementation native to the Dijkstar library. Once all the distances are obtained, the distance matrix will be utilized in all the model formulations.

### 3.4.2 Single Facility Location Problem

The single facility model presented in subsection 3.2 is implemented in Python 3. The implementation is based on a greedy algorithm in that it iterates over each feasible location and calculates the weighted sum to each of the demand facilities. Once all the feasible locations have been evaluated, the algorithm extracts the smallest sum. If there are several locations yielding the same weighted sum, all of these will be returned. Since the algorithm evaluates every possible solution, the algorithm will always yield a global optimum. Furthermore, since the algorithm returns all solutions yielding the lowest objective value the uniqueness is easily evaluated by the algorithm’s output.

### 3.4.3 Heuristic Algorithm

The heuristic presented in subsection 3.3.1 is also implemented in Python 3. The heuristic follows the logic presented in algorithm 3.

### 3.4.4 Two Facility MINLP and MILP

The MINLP formulation is implemented in a combination of Python 3 and GAMS. Since the model is based on the distances obtained from the distance modeling implementation, a program was created which can read a matrix on the format described in subsection 3.4.1 and then create a GAMS input file. The GAMS input file contains the sets to be used as well as the distance matrix and the weight parameters. The file is then complemented by the necessary equations, variables and parameters. The models are then solved through the NEOS Server: State-of-the-Art Solvers for Numerical Optimization. The MILP model is implemented in the same manner as the MINLP.
4 Results

This section will present the results of the single facility implementation on four different test matrices as well as the full map over LKAB seen in Figure 4. Following the single facility implementation, the results of the two facility joint location implementations on the same matrices will also be presented. Lastly, a comparison of the different two facility implementations will be illustrated and commented on.

The test matrices utilized in this section are classified as either being trivial or non trivial. The trivial matrices are constructed in such a way that the optimal solution should be intuitive. These matrices are meant to serve as control problems, to see whether the implementations behave as intended. The non trivial matrices are more challenging problems. These cases are meant to serve as realistic scenarios where the solutions might not be intuitive. The test matrices are of the sizes 5x5 and 10x10. The different sizes is a way of evaluating the scalability of the implementations.

4.1 Single Facility Location Problem

The results of the single facility implementation are shown below. The displayed matrices are the inputs provided to the program and the graphs illustrate how the implementation constructs the graphs. The nodes marked in green contain a demand facility (w [float] in the matrix) and nodes highlighted with a red border indicate the solution to the facility location problem. The solution node and the objective value are stated below the graphs.

4.1.1 Test Cases

\[
\begin{bmatrix}
1 & 1 & 1 & w5 & w100 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
w20 & w5 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Figure 11: Input matrix: "5x5 trivial".
Figure 12: Graph corresponding to Figure 11 Facility placed in Node 5, Objective value 144.349.

\[
\begin{bmatrix}
1 & w_{10} & 1 & 1 & 1 \\
w_{20} & 1 & 1 & w_{10} & 1 \\
1 & 1 & w_{20} & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & w_{10} & 1 & 1 & w_{10}
\end{bmatrix}
\]

Figure 13: Input matrix: “5x5 non trivial”

Figure 14: Graph corresponding to Figure 13 Facility placed in Node 13, Objective value 144.852.
Figure 15: Input matrix "10x10 trivial".

Figure 16: Graph corresponding to Figure 15, Facility placed in Node 29, Objective value 174.852.

Figure 17: Input matrix "10x10 non trivial".
Figure 18: Graph corresponding to Figure 17. Facility placed in either Node 36, Node 46 or Node 56, Objective value 315.562 (Non unique solution).

4.1.2 LKAB

The following matrix is the binary representation of the map shown in Figure 4. The weights are based on the proportions of inventory carried by each demand facility in relation to the total amount of inventory carried by LKAB Kiruna. The result is presented in the same manner as for the test cases, i.e. the solution node is highlighted with a red outline and the objective value stated below the graph.

\[
\begin{bmatrix}
0 & 0 & 0 & 1 & 1 & 1 & w0.249 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & w0.007 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & w0.0540 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & w0.090 & 1 & 1 & 1 & w0.0972 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & w0.180 & 1 & 1 & w0.266 & w0.061 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Figure 19: Input matrix: "LKAB".
4.2 Two Facility Joint Location Problem

The results of the different two facility implementations will be shown below. The input matrices are identical to those used in subsection 4.1. The nodes containing demand facilities are still marked in green. However, the location of facility 1 is marked F1 and the location of the second facility is marked F2. Demands highlighted with a red outline
are served by facility 1 and demands highlighted with a purple outline are served by facility 2. The objective value is still stated below the graphs.

4.2.1 Test Cases

$$\begin{bmatrix}
1 & 1 & 1 & w & 5 & w & 100 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
w & 20 & w & 5 & 1 & 1 & 1 \\
\end{bmatrix}$$

Figure 22: Input matrix: "5x5 trivial".

Figure 23: Graph corresponding to Figure 22. Facility 1 placed in Node 5, Facility 2 placed in Node 21, Objective value 10. The same solution was obtained by all implementations.

$$\begin{bmatrix}
1 & w & 10 & 1 & 1 & 1 \\
w & 20 & 1 & 1 & w & 10 & 1 \\
1 & 1 & w & 20 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & w & 10 & 1 & 1 & w & 10 \\
\end{bmatrix}$$

Figure 24: Input matrix: "5x5 non trivial".
Figure 25: Graph corresponding to Figure 24. Heuristic results: Facility 1 placed in Node 13, Facility 2 placed in Node 6, Objective value 96.568.

Figure 26: Graph corresponding to Figure 24. MINLP and MILP results: Facility 1 placed in Node 13, Facility 2 placed in Node 6, Objective value 86.568.

Figure 27: Input matrix: "10x10 trivial".
Figure 28: Graph corresponding to Figure 27. Facility 1 placed in Node 30, Facility 2 placed in Node 38, Objective value 96.568. The same solution was obtained by all implementations.

![Graph showing placement of facilities and objective value](image)

Figure 29: Input matrix: "10x10 non trivial".

\[
\begin{bmatrix}
  w & 1 & 1 & 1 & 1 & w & 1 & 1 & 1 & w \\
  0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
  1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
  1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
  1 & w & 1 & 1 & 1 & w & 1 & 1 & 1 & w \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]
4.2.2 LKAB

The results of the two facility implementations on the map over LKAB are shown below. The weights used are identical to those in Figure 19. Facility 1 is still marked F1 and facility 2 marked F2. The demand assignment follows the same color code as previously.

Figure 31: Input matrix: "LKAB".
Figure 32: Graph corresponding to Figure 31. Heuristic solution: Facility 1 placed in Node 82, Facility 2 placed in Node 7, Objective value 1.357.

Figure 33: Graph corresponding to Figure 31. MILP solution: Facility 1 placed in Node 82, Facility 2 placed in Node 7, Objective value 1.287.
Figure 34: Graph corresponding to Figure 31. MINLP solution: Facility 1 placed in Node 99, Facility 2 placed in Node 7, Objective value 1.843.

Figure 35: The red squares with an X, correspond to where the facilities are places according to the MILP solution. Since this solution provides the best objective value, only this solutions is illustrated geographically.
4.2.3 Comparison

This section will provide a brief comparison of the results obtained by the different model formulations. The models will be compared in terms of solver, objective value, placement of facilities, demand assignment and computation time.

Test Case: 5x5 trivial matrix

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>MILP</th>
<th>MINLP</th>
<th>MINLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving method</td>
<td>-</td>
<td>Xpress</td>
<td>B &amp; B</td>
</tr>
<tr>
<td>Objective value</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>F1 placed in node</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>F2 placed in node</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Comp. time</td>
<td>0.0028 s</td>
<td>0.043 s</td>
<td>82.53 s</td>
</tr>
<tr>
<td>Iteration count</td>
<td>-</td>
<td>67</td>
<td>2579</td>
</tr>
<tr>
<td>F1 serves demands in node</td>
<td>4 &amp; 5</td>
<td>4 &amp; 5</td>
<td>4 &amp; 5</td>
</tr>
<tr>
<td>F2 serves demands in node</td>
<td>21 &amp; 22</td>
<td>21 &amp; 22</td>
<td>21 &amp; 22</td>
</tr>
</tbody>
</table>

Table 2: Results from the trivial 5x5 matrix (Figure 22).

As shown in Table 2, all the model formulations obtained identical solutions. However, there is a significant difference in computation time. The heuristic algorithm is the fastest, closely followed by the MILP formulation. Both of these implementations are able to solve the problem in well below a second while the MINLP formulation can be solved by ECP in about nine seconds. When using a branch and bound based solver, the computation time for this fairly small and easy problem vastly increases by almost a factor of ten to 82 seconds. In this case, the heuristic or the MILP formulation are the preferred methods for solving the problem.

Test Case: 5x5 non trivial matrix

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>MILP</th>
<th>MINLP</th>
<th>MINLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving method</td>
<td>-</td>
<td>Xpress</td>
<td>B &amp; B</td>
</tr>
<tr>
<td>Objective value</td>
<td>96</td>
<td>86</td>
<td>86</td>
</tr>
<tr>
<td>F1 placed in node</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>F2 placed in node</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Comp. time</td>
<td>0.0018 s</td>
<td>0.563 s</td>
<td>442.25 s</td>
</tr>
<tr>
<td>Iteration count</td>
<td>-</td>
<td>888</td>
<td>10001</td>
</tr>
<tr>
<td>F1 serves demands in node</td>
<td>2, 9, 13, 22 &amp; 25</td>
<td>9, 13, 22 &amp; 25</td>
<td>9, 13, 22 &amp; 25</td>
</tr>
<tr>
<td>F2 serves demands in node</td>
<td>6</td>
<td>2 &amp; 6</td>
<td>2 &amp; 6</td>
</tr>
</tbody>
</table>

Table 3: Results from the non trivial 5x5 matrix (Figure 24).

In this case, a difference in objective value is observed. The heuristic algorithm is not able to find the solution obtained by the MILP and MINLP formulations. The heuristic assigns the demand in node 2 to F1 while the other formulations assign it to F2. Hence, the difference in objective value occurs. Although the heuristic is still the fastest by quite some margin, the result is suboptimal. The MILP formulation is still fast and able to obtain a solution in less than a second. The MINLP formulation is still able to
obtain an equally good solution as the MILP. Although, the computation time is starting to increase to such an extent that it is almost unviable. When using the ECP method for solving the MINLP, the program is not able to go through all iterations before the computation time limit of 1,000 seconds is reached. The MILP formulation is becoming more preferable as the computation time is still fast and it obtains a better solution than the heuristic.

Test Case: 10x10 trivial matrix

<table>
<thead>
<tr>
<th></th>
<th>Heuristic</th>
<th>MINLP</th>
<th>MILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving method</td>
<td>-</td>
<td>ECP</td>
<td>Xpress</td>
</tr>
<tr>
<td>Objective value</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>F1 placed in node</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>F2 placed in node</td>
<td>38</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>Comp. time</td>
<td>0.0098 s</td>
<td>1001.869 s*</td>
<td>1.39 s</td>
</tr>
<tr>
<td>Iteration count</td>
<td>-</td>
<td>647</td>
<td>12717</td>
</tr>
<tr>
<td>F1 serves demands in node</td>
<td>19, 30 &amp; 50</td>
<td>19, 30 &amp; 50</td>
<td>19, 30 &amp; 50</td>
</tr>
<tr>
<td>F2 serves demands in node</td>
<td>17 &amp; 38</td>
<td>17 &amp; 38</td>
<td>17 &amp; 38</td>
</tr>
</tbody>
</table>

Table 4: Results from the trivial 10x10 matrix (Figure 27).

All the models obtain identical solutions. The heuristic being the fastest followed by the MILP. The MINLP reaches it’s computation time limit of 1,000 seconds. Note that the MINLP can no longer be solved using the Branch and Bound based solver as the size of the problem is too large. The MILP formulation is so far the most consistent, it finds the best solutions and it is quick. The MINLP is not able to compete with the MILP or the heuristic as it is too slow.

Test Case: 10x10 non trivial matrix

<table>
<thead>
<tr>
<th></th>
<th>Heuristic</th>
<th>MINLP</th>
<th>MILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving method</td>
<td>-</td>
<td>ECP</td>
<td>Xpress</td>
</tr>
<tr>
<td>Objective value</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>F1 placed in node</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>F2 placed in node</td>
<td>76</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td>Comp. time</td>
<td>0.0146 s</td>
<td>1001.962 s*</td>
<td>25.237 s</td>
</tr>
<tr>
<td>Iteration count</td>
<td>-</td>
<td>590</td>
<td>155396</td>
</tr>
<tr>
<td>F1 serves demands in node</td>
<td>1, 6 &amp; 10</td>
<td>1, 6 &amp; 10</td>
<td>1, 6 &amp; 10</td>
</tr>
<tr>
<td>F2 serves demands in node</td>
<td>72, 76 &amp; 80</td>
<td>72, 76 &amp; 80</td>
<td>72, 76 &amp; 80</td>
</tr>
</tbody>
</table>

Table 5: Results from the non trivial 10x10 matrix (Figure 29).

Once again, all the models obtain identical solutions. The heuristic is sustaining it’s calculation time advantage, being able to solve the problem in a fraction of a second. The MILP on the other hand has had a significant increase in computation time compared to the previous cases. It is still fast in comparison to the MINLP, which is not able solve the problem in less than 1,000 seconds.
The LKAB matrix provides some interesting results as all three models obtain differing solutions. The MILP model finds the best solution in terms of objective value, and it manages to do so in under a minute. The MINLP model times out at 1,000 seconds as previously and does not find a particularly good solution. The heuristic is still fast and manages even this 12x17 matrix in well below a second. The objective value obtained by the heuristic is slightly worse than the MILP but a bit better than the MINLP.
5 Discussion

The discussion will deal with the topics of optimality, the application of a grid and the graph implementation as a means of modeling distance, evaluation of the different approaches to the two facility joint location problem and finally, some potential model improvements and further research.

5.1 Optimality

Regarding the single facility model presented in subsection 3.2 and the results obtained by the implementation showed in subsection 4.1, optimality can be guaranteed. The reason being that the model is quite simple, it is a mixed integer linear programming model with a single linear constraint. Further, the implementation of the model is an iterative function which evaluates every feasible solution. Given that all possible solutions are evaluated, the function simply compares all the possible locations and chooses the location yielding the lowest possible objective value. Hence, the resulting solution will always be optimal. However, the solution may not be unique depending on the problem. As illustrated in Figure 18 there are three potential locations, all yielding the same objective value of 315.562. In this case when the solution is non-unique, the model does not provide any additional insight as to which solution might be practically better. This will be further discussed in subsection 5.4.

When looking at the two facility models, the possibility of obtaining optimal solutions is present within all the models. However, as shown by the results in subsection 4.2, the objective value obtained by the different models can vary when applied to the same test case. The heuristic algorithm from subsection 3.3.1 can not guarantee optimal solutions. This is due to the algorithm not evaluating every feasible solution. If the scenario arises where several demands are at an equal weighted distance away from the current location of facility 1, the algorithm simply chooses one of these demands to reassign. Hence, the algorithm is inherently suboptimal. Although, the algorithm has the benefit of being extremely fast computationally. None of the input matrices shown throughout this report yielded a computation time above one second. In other words, the heuristic is fast and in some cases will provide an optimal solution. However, in the general case, the results obtained by the heuristic should not be considered optimal.

The mixed integer non linear formulation is also able to yield optimal results depending on the solver used during the implementation. The solvers are stated in the comparison section, subsection 4.2. When using a solving method as ECP or Branch and Bound, the obtained solutions should be considered optimal as these are exact methods for solving mixed integer problems [15]. However, since MINLP problems are NP-hard, the computation time needed for the test cases often exceeded the time limit of 1,000 seconds. Due to this time limit, the solvers were in some cases terminated before an optimal solution could be guaranteed. Therefore, the observed results of the MINLP model sometimes differed from MILP model. This is particularly noticeable when both models were applied to the LKAB matrix in Table 6. In other words, as the size of the problem increases and a time exceeding the time limit of 1,000 seconds is needed, the MINLP model will not be able to guarantee optimal solutions. Only when the solver is terminated before the time limit can the results be considered optimal e.g. Table 2.

In comparison to the inconsistent results of the heuristic and the MINLP formulation,
the mixed integer linear programming model proved to be consistently the best model. The MILP model was able to obtain an optimal solution to all the test cases as well as the LKAB matrix. The MILP model was solved using the Xpress solver in GAMS which is based on a Branch and Bound solving method for mixed integer models. In each of the tests the solver was terminated due to fulfilling the absolute optimality criterion, i.e. \(|PB - DB| = 0\) where PB is the objective value of the current best feasible solution and DB is the current lower bound on the optimal value. Therefore, the solutions obtained by the MILP model are to be considered optimal solutions. This is also consistent with the MILP model always providing the best results throughout the testing.

5.2 Distance Modeling and Implementation

The distance modeling method of utilizing an application of a grid over a geographic area, to then translate each square of the grid into a node in a graph proved useful. Modeling distance this way provides great flexibility, the model can be adjusted and modified after any particular specifications. In the application to LKAB the choice was made to make the squares fairly large, about 100x100 m². This was done to provide a bit more freedom for the placement. In other words, the aim was not to pinpoint the exact coordinates for the center of the facility, but rather provide a general area. Due to the size of these squares, the assumption of it being possible to place a facility within the same square as a demand is not unreasonable. Most, if not all of the squares contain extra space to allow the placement of the facility. Furthermore, there are different types of facilities available, some of which allow for closer proximity to a building. This means that, if the highest level of security facility is built, the proximity to a building can be as close as a few meters. However, for a different application that might not be the case. The implementation can then be adjusted to not include any nodes containing demands as feasible but still create connections to neighbouring nodes. Hence, the model itself and the implementation is highly adaptable. Furthermore, the size of the squares in the grid can also be increased or decreased in size, resulting in a differing amount of nodes within the corresponding graph. An increase of nodes (decrease in size of the squares) would mean that the distances traveled would be more accurately represented, at the expense of a larger problem, i.e. more calculations and some loss in flexibility of the recommended placement area. Another advantage of this modeling approach is that many otherwise difficult to formulate constraints can be incorporated in the construction of the graph. In the case of LKAB, squares containing large buildings, offices or inaccessible routes could be directly excluded from the graph by not creating any connecting edges to these. Hence, no proximity or infeasibility constraints had to be explicitly formulated in the minimization models.

The implementation of the distance modeling utilized Dijkstra’s algorithm to obtain the shortest paths from any given origin node to a destination node. In the context of this thesis, the destinations are defined as demands. The choice to use Dijkstra’s algorithm was case specific. As Dijkstra is a fast algorithm which works with both directed and undirected graphs, it was deemed a good fit for this application. The downside of Dijkstra is that it can not be used when the graph contains negative edge weights. This of course is not a problem in this thesis as there are no reasons for negative weights. However, in a different application the need for negative weights might arise. The distance modeling proposed could still be successful, although a different shortest path algorithm such as the Bellman Ford should be used. Further improvements could also
be made to increase the performance of Dijkstra’s algorithm, such as implementing metaheuristics. This could further increase the scalability of the model [16].

The distance modeling allows for many advantages and flexible implementations. However, a significant disadvantage is that it is tedious. Since everything is currently being made by hand when constructing the input matrices, the model is prone to human errors. Furthermore, it is also important to note that this way of modeling distance has its limitations. As the area being modeled becomes substantially large, the squares within the grid need to be quite small for this method to yield any significant results. If the distances become large enough it might be easier to use a different distance measure such as the euclidean distance or large circle distance. The usage for this type of modeling could however be viable in large contexts, e.g. if one wishes to model road networks.

5.3 Two Facility Joint Location Problem

The two facility extension to the original facility location problem yielded three different models. The models performed with differing results as seen in section 4.

The first model to be formulated was the heuristic algorithm. The algorithm proved to be efficient and able to handle larger problems without exceeding one second of computation time. The results of the heuristic were sometimes suboptimal due to the implementation. This is an inherent flaw of the algorithm. In order to evaluate and compare the results of the algorithm, the MINLP model was developed. The MINLP model proved that the heuristic is unreliable and that some of the solutions provided by the heuristic could be improved. The problem with the MINLP model is that it is difficult to solve. The computation time needed for fairly small problems rendered the MINLP model unviable. Because of this, a linearization of the MINLP model was developed, i.e. the MILP model. The MILP model proved efficient enough. Although the computation time could not match the heuristic, the obtained solutions were overall better. The MILP model is deemed to be the preferred model by the authors as it was able to obtain optimal solutions during the testing. The MILP model was also able to obtain an optimal solution to the LKAB problem within one minute of computation time.

To conclude, the models were developed sequentially to continuously improve the results. The last iteration, i.e. the MILP model proved to be the best model. That conclusion was drawn as it is moderately fast computationally and able to provide optimal solutions to the largest problem (the LKAB map).

5.4 Model Improvements and Further Research

The models presented in this thesis could certainly be further improved to provide better results and make the models directly applicable to different cases. Even though the purpose of the thesis was to solve a single facility location problem, improvements will be presented to allow for further applications. Starting with the distance modeling. As mentioned in subsection 5.2, Dijkstra’s algorithm could be replaced with the Bellman Ford algorithm in order to enable the use of negative edge weights. Other improvements previously discussed is the utilization of metaheuristics as a complement to the shortest path algorithm to improve the scalability. Regarding future research, machine
learning could be an interesting way of analyzing maps. A well formulated machine
learning algorithm could perhaps recognize buildings and inaccessible parts of an area
and thereby create the graph automatically. The implementation of such an algorithm
could reduce the risk of human error when constructing input matrices.

The minimization models for the single facility as well as the two facility problem
could also be improved. The current models are only considering a weight associated
with a demand and the distance between the demand and the new facility. In order to
improve the model, several new parameters could be introduced in order to account
for other types of costs associated with the placement. Some locations may need a
larger amount of preparation work. Depending on the type of facility to be built, water
lines or electricity connectivity may have to be constructed prior to the facility being
built. Hence, a larger cost is incurred. Therefore, a cost parameter is suggested to
be introduced to account for the different increases in cost some areas may incur. A
different scenario could be that some areas are more beneficial in terms of offering
some additional utility. The increased value of having a facility in that area could be
denoted with a negative cost. Let this cost parameter be called $c_i$. Further, the current
models do not incorporate any decisions in terms of the amount of facilities to be built.
Therefore, an additional improvement could be to introduce a fixed cost of building the
facility. This cost would then be assumed to be universal for each facility and location.
Let this cost be denoted $F$. The addition of this constant would allow the models to
choose the optimal amount of facilities to be placed. Given the introduction of more
information to the model, a better informed decision could be made. However, it may
not be fully reasonable to assume that the budget is infinite and therefore the models
should not allow for the building of an unrestricted amount of facilities. This could
be remedied by either introducing a constraint limiting the amount of facilities, or by
introducing a budget variable $B$. The introduction of the budget variable along with
a constraint ensuring that the incurred costs of building the facilities do not exceed
the budget could solve this problem. Incorporating these improvements, a suggestion
model is shown below.

<table>
<thead>
<tr>
<th>Set</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$i$</td>
<td>Set of feasible nodes $i$ in graph</td>
</tr>
<tr>
<td>$K$</td>
<td>$k$</td>
<td>Nodes $k$ containing a demand facility, $K \subset I$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Param</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_k$</td>
<td>$\mathbb{R}$</td>
<td>Weight associated with demand $k$</td>
</tr>
<tr>
<td>$a_{ik}$</td>
<td>$\mathbb{R}$</td>
<td>Distance from node $i$ to demand $k$</td>
</tr>
<tr>
<td>$c_i$</td>
<td>$\mathbb{R}$</td>
<td>Additional cost of placing a facility in node $i$</td>
</tr>
<tr>
<td>$F$</td>
<td>$\mathbb{R}$</td>
<td>Fixed construction cost for a facility</td>
</tr>
<tr>
<td>$B$</td>
<td>$\mathbb{R}$</td>
<td>Maximum spending budget</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>Binary</td>
<td>Facility placed in node $i$</td>
</tr>
<tr>
<td>$y_{ik}$</td>
<td>Binary</td>
<td>Facility in node $i$ serves demand $k$</td>
</tr>
</tbody>
</table>

$$y_{ik} = \begin{cases} 1 & \text{when facility in node } i \text{ serves demand } k \\ 0 & \text{else} \end{cases} \quad (13)$$
Minimize: \[ \sum_{i \in I} Fx_i + \sum_{i \in I} \sum_{k \in K} y_{ik} w_{ik} a_{ik} c_i \] (14a)

Subject to:
\[ \sum_{i \in I} x_i \geq 1 \] (14b)
\[ \sum_{i \in I} x_i (F + c_i) \leq B \] (14c)
\[ \sum_{i \in I} y_{ik} = 1 \quad \forall k \] (14d)
\[ y_{ik} \leq x_i \quad \forall i, k \] (14e)

The improved model shares many similarities to the previously shown models in section 3. The difference being the introduced variables \( B, F \) and \( c_i \). The constraints are also quite similar except for Equation 14c which ensures the budget is not exceeded. The constraint Equation 14b ensures at least one facility is placed. Equation 14d ensures every demand is served by one facility. Lastly, the constraint Equation 14e states that a demand in \( k \) can only be serviced from node \( i \) if there is a facility located in \( i \).

The reason this model was not implemented is simply the lack of information regarding the additional variables.
6 Conclusions

The thesis set out to solve a facility location problem within a confined space where common distance approximations would not provide accurate results. The location problem would also include proximity and path constraints. Hence, the research questions were: How to solve the facility location problem in the presence of path and location constraints? The sub-problems associated with the research questions were: How to model distances and paths in a confined area with path and location constraints?

The distance and path modeling as well as the feasible and unfeasible area modeling can be solved by utilizing a graph representation of a geographic area. Each node in the graph represents a small section of the area and inaccessible areas or areas too close to inaccessible areas are left out of the graph. Thereby, the graph contains gaps where traversing is not possible. The gaps in the graph ensure that the path restrictions, proximity constraints and unfeasible areas were accounted for when formulating an optimization problem. The distances between feasible areas and destination areas, i.e. demands, could then be obtained by a shortest path algorithm. In this case, Dijkstra’s algorithm was implemented. Once the distances could be modeled, the next step was to create an optimization model for choosing the best location to build a new facility.

The facility location problem was solved using a mixed integer linear programming formulation. The initial model would only be viable for obtaining the best location for a single facility. Therefore, an extension to the model was made to solve the problem of placing two facilities. The two facility joint location problem was solved in three different manners, a heuristic algorithm, a mixed integer non linear formulation and a mixed integer linear formulation. Testing of the models concluded that the most viable and consistent model was the mixed integer linear formulation, as this model provided optimal solutions in all tests and was able to do so requiring a modest amount of computation time.

The final conclusions drawn from the models are that there is still room for improvement. The single facility and the two facility models could be integrated with the introduction of more parameters. The model would then be able to account for more managerial decisions, e.g. how many facilities should be built given a certain budget. The distance modeling could also be made more versatile by implementing other shortest path algorithms to enable the use of negative edge weights. Lastly, some interesting future research could be made into metaheuristics and machine learning to further improve the distance modeling and simplify the process of constructing a graph.
References


