ROBUST MAXIMUM COVERING LOCATION PROBLEM (RMCLP)

SAEID JAFARIPOUR
Master of Computer Science, Kharazmi University, 2019

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

ROBUST MAXIMUM COVERING LOCATION PROBLEM (RMCLP)

SAEID JAFARIPOUR

Date of Defence: April 24, 2023

Dr. Robert Benkoczi Thesis Supervisor	Professor	Ph.D.
Dr. Wendy Osborn Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. Amir Akbary-Majdabadno Thesis Examination Committee Member	Professor	Ph.D.
Dr. John Sheriff Chair, Thesis Examination Committee	Assistant Professor	Ph.D.

Dedication

I dedicate this thesis to my wonderful parents.

Abstract

The Maximum Covering Location Problem (MCLP) is a widely recognized optimization problem used in facility location planning. The objective of this problem is to minimize costs while maximizing accessibility to customers. In this thesis, the MCLP is being solved as an optimization problem under uncertain customer benefits in a network, to minimize regret, which is the difference between the cost of the optimal solution under the worst-case scenario and the cost of the current solution under the worst-case scenario.

Three algorithms were implemented to solve the problem and find the optimal solution, including an exact algorithm and two approximate algorithms. The algorithms were evaluated using various instances, including the OR-Library and randomly generated instances. The results indicate that the exact algorithm is better at minimizing regret, but it is unable to solve large instances within the allotted time limit. Also, one of the approximate algorithms based on a Mean-Scenario, which is a 2-Approximation general algorithm, indicates very competitive results to obtain the Min-Max Regret. The observations of this thesis confirm the results of related research for both the exact algorithm and the Mean-Scenario algorithm, which rely on standard and general methodologies. Also, the solutions obtained by the other approximate algorithm based on randomized rounding are within only nine percent of the Mean-Scenrio algorithm results, which proves the value of this approach.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Robert Benkoczi, for his invaluable guidance and support throughout the duration of this project. His patience, encouragement, and expertise have been instrumental in helping me complete my master's thesis.

I would also like to thank the members of my thesis committee, Dr. Wendy Osborn and Dr. Amir Akbary-Majdabadno, for their constructive feedback and valuable insights. I am grateful for the opportunity to have received such comprehensive guidance from such esteemed scholars.

I would like to extend my appreciation to my friends and family for their unwavering support and encouragement. Their belief in me kept me motivated and inspired when I doubted myself.

Finally, I would like to express my gratitude to the University of Lethbridge for providing me with the resources and opportunities to pursue my studies and complete this project.

Thank you all for your invaluable contribution to my academic journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The problem of Maximal Covering Location (MCLP) is a location-allocation problem that requires finding the best placement of facilities to cover the customers within the facilities' coverage radius. Customers located within this radius will receive coverage. The objective is to maximize the number of customers covered while minimizing the cost of opening and operating the facilities [1]. In the certain MCLP, every customer possesses a positive benefit value. However, in the uncertain MCLP, which is explored in this thesis, the benefit value for each customer falls within a range. Any valid combination of values for the benefit determines a scenario for the uncertain MCLP.

The regret of a scenario is defined as the difference between the cost of the optimal solution for the scenario and the cost of the chosen solution [2]. In the Min-Max Regret approach for MCLP, the goal is to find a solution that minimizes the maximum regret of the decision-maker over all possible scenarios. In other words, the decision maker wants to minimize the maximum possible loss due to choosing a sub-optimal solution. The Min-Max Regret MCLP model involves two stages. In the first stage, a set of potential facility locations is identified. In the second stage, the decision maker selects a subset of facilities to be opened based on the uncertainty about the customer benefits.

The Min-Max Regret MCLP model can be formulated as a mixed-integer linear programming (MILP) problem [3]. The objective function minimizes the maximum regret, and the constraints ensure that at least one facility covers each customer's demand. The solution of the Min-Max Regret MCLP model identifies a given number of facility locations

and the corresponding customer assignments that minimize the maximum regret.

Overall, the Min-Max Regret approach for MCLP is helpful for decision-makers concerned about making sub-optimal decisions when faced with uncertainties in problem input. By minimizing the maximum regret, decision-makers can identify robust solutions less sensitive to changes in demand and other uncertain factors.

1.1 Research Overview and Motivation

The Min-Max Regret approach aims to reduce the risk of making decisions based on minimizing the regret. The method allows to balance the trade-off between expected performance and the potential losses associated with sub-optimal facility location decisions. The central goal of this thesis is to tackle the uncertain variant of MCLP, with the ultimate objective of achieving the Min-Max Regret. To achieve this objective, exact and approximate algorithms are utilized to obtain solutions to the problem. Furthermore, the effectiveness of these algorithms is also assessed through an evaluation of their efficiency.

1.2 Contributions

Much research has been conducted on optimization under uncertain conditions, with or without considering the Min-Max Regret. A wide range of problems has been investigated in this area.

The MCLP aspect of this thesis takes inspiration from research by Church et al. [1] on the maximal covering location problem. However, this thesis differs by considering the benefits of uncertainty for customers, which introduces the Min-Max Regret criteria to the problem. The research by Coco et al. [4] is closely related to this thesis, as it focuses on a robust version of the MCLP that seeks to achieve the Min-Max Regret criterion. While the authors aim to address the entire customer demand, this thesis focuses on maximizing the coverage of the customers to obtain the Min-Max Regret.

The main contribution of this thesis is developing an approximate algorithm based on

randomized rounding called *AAF*, which deals with generating a set of inequalities to obtain the Min-Max Regret in MCLP. The *AAF* algorithm consists of a master problem and a sub-problem, which solve the problem iteratively. In each iteration, the sub-problem aims to find a better solution compared to the current solution provided by the master problem by adding a new constraint to it. These constraints form a set of inequality constraints, which are used to obtain the Min-Max Regret. The constraints generated by *AAF* show potential for achieving results similar to the 2-Approximation algorithm known as *AAM*, which is also studied in this thesis. This offers a clear direction to use the concept of *AAM* in the approximate algorithm based on randomized rounding *AAF* to improve this algorithm for future studies.

Chapter 2

Problem Definition and Related Work

This chapter outlines the importance and the standard approaches to the problem, as well as the definition of MCLP in more detail. To illustrate the process of solving MCLP, two examples are presented: one for the certain MCLP and another for the uncertain case. Additionally, the chapter provides a review of the relevant literature.

2.1 Standard Approaches and the Importance of the MCLP

The maximum covering location problem is a type of optimization problem that aims to maximize the coverage of customers by selecting a subset of locations from a given set of candidate locations. Depending on the kind of problem, the coverage of a location can be defined in different ways. According to Church et al. [1], the objective of MCLP is maximizing the covered population within range called coverage radius.

The MCLP is a fundamental optimization problem in various fields, including computer science, operations research, and geographic information systems. In computer science, it is used to allocate facilities or resources to maximize their coverage. For example, it can help design wireless network access points in a building to cover the maximum number of users [5]. In operations research, the problem is used to allocate resources or facilities to maximize efficiency. For example, it can assist in specifying the optimal locations for warehouses or distribution centers or selecting the best locations for service centers or repair facilities in a maintenance network [6]. Moreover, the maximum covering location problem is utilized in geographic information systems to model the optimal locations for a specific

objective. For instance, it can help select the best sites for new retail stores or emergency facilities to serve the maximum number of people [16]. Despite the different applications, the common objective of the problem is to determine the optimal locations that can provide the best coverage or service to the customers [7]. Overall, the maximum covering location problem is crucial because it helps organizations make effective decisions about allocating resources to maximize their coverage.

Several approaches can be used to solve this problem, such as exact algorithms, heuristics, and approximation algorithms. Exact algorithms guarantee that the solution found is globally optimal, meaning it provides the best possible solution to the MCLP. However, they can be computationally expensive and time-consuming, especially for larg problem instances. The heuristic algorithms provide solutions quickly but without guaranteeing optimality. In other words, the heuristic algorithms achieve a balance between finding a good solution and finding it in a reasonable amount of time, but the gap between the obtained solution and the optimal solution may be substantial in certain cases.

Approximation algorithms provide solutions that are guaranteed to be within a particular factor of the optimal solution, and they are often faster than the eaxct algorithms. Still, they may not provide solutions as good as those found by exact algorithms, and the approximation factor may not be tight enough for some applications.

An aspect of the MCLP is NP-hardness, meaning that it is one of the most challenging problems and is difficult to solve efficiently. In general, NP-hard problems are challenging problems in the NP complexity class, as they require exponential time to solve and are at least as difficult as the hardest problems in NP. Polynomial time refers to the running time of the algorithm growing no faster than a polynomial function of the input size. This demonstrates that finding an optimal solution to MCLP is computationally challenging and unlikely to be solvable in polynomial time, especially for large-scale instances. As a result, approximation algorithms and heuristics are often used to find sub-optimal solutions to MCLP. Developing these algorithms helps to solve large-scale MCLP instances but does

not always find the optimal global solution [8].

2.2 MCLP with Certain Values

This section discusses the certain MCLP. As dicussed before, each customer is assigned a positive benefit value, and a facility's coverage range is determined by its service radius. The subsequent part of this section provides an example of the cetain version of MCLP.

A set of known customers represented by the index set $I = \{1, 2, ..., n\}$ is given, as well as a set of candidate locations for the facilities, represented by the index set $J = \{1, 2, ..., m\}$. The task is to compute the location for a given number of T facilities so that the customer benefit served by these facilities is maximized.

A certain MCLP example was examined, where the benefit value for each customer is assumed to be fixed. According to Figure 2.1, a line containing three customers is considered. Each customer i has a positive value of benefit defined by b_i. For this example, the values of b_i are evaluated as follows: b_i = [5, 10, 1] for the three customers. Only one facility is supposed to be placed in such a way as to cover the maximum possible benefit for the customers. The coverage radius is chosen to be one, since we want the facility to be close to the customers. So first, three circles with a coverage radius of one centered by points A, B, and C, are placed to divide the line into five segments. The segments identified as x_1, x_2, x_3, x_4, and x_5 are the potential candidates for placing one facility.

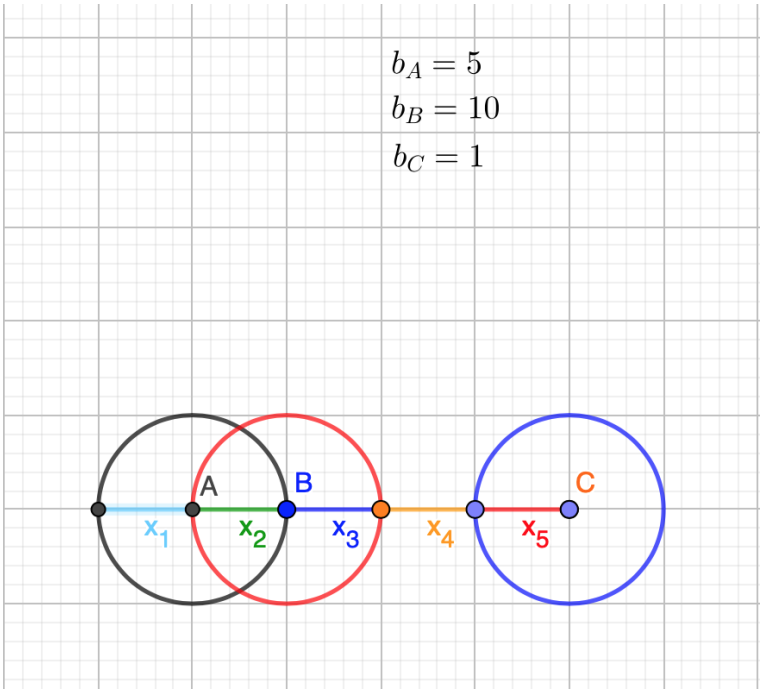

Figure 2.1: Deterministic MCLP on the line

An Integer Programming model is written to find the maximum benefit by locating one facility on the line.

$$\textbf{max } 5z_A + 10z_B + z_C \tag{2.1a}$$

$$\textbf{subject to} \quad z_A \leq x_1 + x_2 \tag{2.1b}$$

$$z_B \leq x_2 + x_3 \tag{2.1c}$$

$$z_C \leq x_5 \tag{2.1d}$$

$$x_1 + x_2 + x_3 + x_5 = 1 \tag{2.1e}$$

$$x_j, z_i \in \{0,1\} \tag{2.1f}$$

The variable z determines whether a customer is covered, while the variable x determines whether a facility is selected. If a customer is covered, the value of z is one, and if not it is zero. Also, if a facility is placed, the value of x is one, otherwise, it is zero.

The formulation includes constraints based on the coverage range of each facility. Customer A falls within the coverage range of the candidate locations x_1 and x_2, which constructs the constraint $z_A \leq x_1 + x_2$. The purpose of this constraint is to determine that customer A can be served by the two candidate facility locations x_1 and x_2. Customer B is covered by x_2 and x_3, which means this customer can be served by the candidate locations x_2 and x_3. Accordingly, the constraint $z_B \leq x_1 + x_2$ is constructed. Also, customer C is solely within the range of x_5 and it only gets served by this candidate location, which shows the purpose of the constraint $z_C \leq x_5$. The last constraint in the formulation indicates that the sum of x_1, x_2, x_3, and x_5 is one because only one facility will be placed. Since the candidate location x_4 does not fall within the coverage range of any customers, it is excluded from consideration. According to Figure 2.1, and the provided formulation supported by the constraints explanation, it is proven that placing one facility at the candidate location x_2 can cover a maximum benefit of 15.

If we assume that I_3 is an identity square matrix with all diagonal elements equal to one and all other elements equal to zero, we can express the constraints (2.1b)-(2.1f) in matrix form as follows: $I_3 z - [1\ 1\ 0\ 0;\ 0\ \ 1\ 1\ 0;\ 0\ 0\ 0\ 1]x \leq 0$. We denote matrix $[1\ 1\ 0\ 0;\ 0\ \ 1\ 1\ 0;\ 0\ 0\ 0\ 1]$ by A, which is called the coverage matrix and is defined as follows: the values of a_{ij} are 1 or 0, depending on the coverage of customer i by facility j.

$$a_{ij} = \begin{cases} 1, & \text{If customer } i \text{ is covered by facility } j \\ 0, & \text{Otherwise} \end{cases}$$

For example, if a facility is placed anywhere on segment x_1, it covers customer 1; if it is placed anywhere on segment x_2, it covers 1 and 2. So, the value for a_{11} is 1, for a_{12} is 1, for a_{13} is 0, and for a_{14} is 0, and so on. Also, the entries have 12 values. The size of the identity matrix I and the coverage matrix A depends on the number of customers and the number of candidate locations. In this example, we have three customers, so I_3 is chosen. Also, the size of A is $n \times m$, which is 3×4 in this example. Here, n is the number of customers and

m is the number of candidate locations, as discussed earlier.

The customers shown as A, B, and C in Figure 2.1 are identified as the rows of the coverage matrix A, and the four candidate locations are recognized as the columns of the matrix.

The general formulation for the certain version of MCLP can be derived from the example. As mentioned before, $I = \{1,2,...,n\}$ is a given set of the customres and $J = \{1,2,...,m\}$ is the set of candidate locations. The formulation for MCLP is as follows:

$$\textbf{max} \sum_{i \in I} b_i z_i \tag{2.2a}$$

$$\textbf{subject to} \quad z_i \leq \sum_{j \in \mathbf{J}} a_{ij} x_j \quad \forall i \in I \tag{2.2b}$$

$$\sum_{j \in \mathbf{J}} x_j \leq T \tag{2.2c}$$

$$x_j, z_i \in \{0,1\} \tag{2.2d}$$

2.3 MCLP with Uncertain Values

In the problem of this thesis, the customers' benefit, b_i, is uncertain. Accordingly, the value of the benefit is situated between known lower and upper bounds $[b_i^-, b_i^+]$. An example will be discussed in the remainder of this section. A scenario refers to any assignment of specific values in the allowed interval to all benefits. As the interval comprises an infinite range of values, the number of scenarios is also infinite. Each scenario has an optimal solution. The difference between the cost of a given solution under a scenario and the cos of the optimal solution under the scenario is called regret, according to Averbakh et al. [9]. Given a solution, the scenario that maximizes the regret for that solution is the worst-case scenario for the solution. Among the infinite set of scenarios, we are interested in the worst-case scenarios. The approach for choosing the worst-case scenarios will be discussed in Chapter 3.

The solution that we need to compute is fixed and is used with any of the possible scenarios. The task is to compute the location of T facilities to minimize the maximum regret. As discussed earlier, given a solution Z, there is a worst-case scenario where the regret is maximum. Therefore, we want a solution Z^* to minimize this maximum regret.

An example of MCLP with uncertain values of benefits is provided below. The following intervals define the benefits for the customers:

Benefit for customer 1 = [5, 10]

Benefit for customer 2 = [4, 8]

Benefit for customer 3 = [3, 20]

Considering the extreme values within the benefit intervals, the following scenarios address the Min-Max Regret for this example.

S_1 = [5, 4, 3], S_2 = [5, 4, 20],

S_3 = [5, 8, 3], S_4 = [5, 8, 20],

S_5 = [10, 4, 3], S_6 = [10, 4, 20],

S_7 = [10, 8, 3], S_8 = [10, 8, 20].

The rows of Table 2.1 are identified as x_1, x_2, x_3, and x_5. These are the candidate locations for opening a single facility $T = 1$ on the line, based on the Figure 2.1 so the maximum regret is minimized. Each column in the table represents a scenario. I note that the table illustrated here has a number of columns that is exponential in the number of customers.

The values of Table 2.1 answer the problem according to each scenario on each candidate location. To fill out Table 2.1, Figure 2.1 could be called for convenience. As an example, x_1 is only covering customer A. Thus, for any of the eight scenarios, the answer

for that row is the benefit of customer A. Accordingly, the values of benefits in the eight scenarios under each candidate location are written in Table 2.1.

Table 2.1: The Min-Max Regret table.

-	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	Max regret
x_1	5	5	5	5	10	10	10	10	15
x_2	9	9	13	13	14	14	18	18	11*
x_3	4	4	8	8	4	4	8	8	16
x_5	3	20	3	20	3	20	3	20	15
Max answer	9	20	13	20	14	20	18	20	-

The maximum value of each column, the optimal solution under the corresponding scenario, is written in the last row of the table and is called Max-answer. For each column, the difference between the Max-answer row with each of the x_1, x_2, x_3, and x_5 rows is the benefit we lose if we commit to that solution under that scenario. This difference is the regret we aim to compute. The maximum lost benefits under each solution, x_1, x_2, x_3, and x_5, is written in the last column, Max regret. Among those four values for Max regret, the minimum is 11, highlighted by a star sign. Consequently, the Min-Max Regret for this example is 11. Therefore, the optimal solution is x_2 under S_2. This solution is robust since it stays fixed under the different scenarios.

2.4 Literature Review

One of the oldest papers about the maximum covering location problem is done by Church et al. [1]. The authors worked on a new approach to solve the MCLP using integer programming. They show the validity of their approach with different test instances through synthetic and real-world data sets. In more detail, the authors consider a service distance S and a fixed number of facilities to be located in such a way as to maximize the

population being served along distance S. There is also a mandatory coverage distance T, which $S \leq T$. The authors' goal is firstly, to maximize the population covered within a distance of S, and secondly, to ensure that no individual is more than T units away from their closest facility. This paper uses two techniques for finding the solutions: Heuristic approaches and linear programming. The main contribution of [1] is that the authors developed a new integer programming formulation for MCLP that is compact and effective by using a combination of techniques. To make the formulation compact, they used a set of binary variables to represent each facility's coverage rather than a separate constraint for each customer-facility pair. In terms of effectiveness, they incorporated a set of inequalities and heuristic algorithms that strengthen the relaxation of the problem to help the solution process. The authors also provide insights into the structure of optimal solutions, which may help develop efficient heuristics and approximation algorithms. In this thesis, we are using linear programming techniques to solve the MCLP. The main difference is that this paper does not discuss uncertainty regarding the customers, which means the corresponding values to the benefits of their problem are single values.

Another optimization algorithm for the MCLP with fuzzy coverage radii is proposed by Davari et al. [10]. The coverage radii are modeled as fuzzy numbers in this paper. The algorithm proposed in the paper is a greedy algorithm, which is a two-step procedure that generates a solution and then modifies it using variable neighborhood search to improve the quality of the solution. The authors evaluated their work with different sizes and complexity. The results showed that their algorithm worked better than the existing algorithms regarding solution quality and computational efficiency.

A survey paper is done by Snyder et al. [11] to gain a clearer insight into the robust location problems. In the paper, the authors have discussed the different types of uncertainty which can affect the maximum covering location problem, such as demand uncertainty, supply uncertainty, and a combination of these uncertainties. Since this is a review paper, the author addresses the models covering these uncertainties, such as stochastic program-

ming, robust optimization, and scenario-based approaches. The authors also discuss the limitations of these approaches, such as computational complexity, difficulty in modeling uncertainty, and limitations in the types of uncertainty. According to this paper, one of the common measures for robust problems is the Min-Max Regret, defined as the difference between the cost of the solution in a given scenario and the cost of the optimal solution for the same scenario. To calculate the solution under uncertain values of benefits within an interval, the techniques for regret maximizing scenarios rely on the scenario, typically having each parameter set to an endpoint of its interval. The authors also evaluate the performance of the discussed models using a set of test problems.

According to a paper by Coco et al. [4] the authors aim to cover the total demand of the customers in a network using two algorithms: an exact algorithm and an approximate one. This paper addresses a problem similar to this thesis's problem of solving the Maximal Covering Location Problem (MCLP) by minimizing the maximum regret. However, there are some differences between the two problems. In this paper, the authors aim to cover all customers with a set of at most T facilities with a maximum total benefit of the facilities. In contrast, the problem in this thesis aims to cover the maximum possible customers' benefits only to obtain Min-Max Regret. Another difference between the two problems is how the uncertain benefits are assigned. In this thesis's problem, each customer has an interval of benefits, whereas, in [4], the benefits are associated with the facilities.

A robust version of the Min-Max Regret problem is explored by Coco et al. [12]. This article addresses two Min-Max Regret covering problems: the Min-Max Regret Weighted Set Covering Problem (Min-Max Regret WSCP) and the Min-Max Regret Maximum Benefit Set Covering Problem (Min-Max Regret MSCP). These problems are considered robust optimization counterparts of the Weighted Set Covering Problem and the Maximum Benefit Set Covering Problem. In both problems, uncertainty in the data is modeled by utilizing an interval of continuous values that represents all possible values that every uncertain parameter can assume. The study has several significant contributions, including a mathematical

formulation for the Min-Max Regret MSCP and exact and heuristic algorithms for the Min-Max Regret WSCP and MSCP. Five heuristic algorithms, as discussed in Section 2.1, are applied for both problems, including two scenario-based heuristics, a path relinking, a pilot method, and an integer-based programming heuristic. For example, path relinking uses information from a set of previously generated solutions to improve the quality of the current solution. The algorithm creates a path of iteratively improved solutions until a high-quality solution is found. The primary objective is to analyze the impact of these methods on handling robust covering problems, taking into account both solution quality and performance.

Another research by Kasperski et al. [3] considers uncertainty and regret. The authors propose a heuristic algorithm for solving a specific class of combinatorial optimization problems called the Interval Data Min-Max Regret Combinatorial Optimization Problem (IDMR-COP). The IDMR-COP involves finding a subset of items from a given set that satisfies certain constraints and minimizes the maximum regret over a set of intervals. The authors propose a heuristic algorithm with a guaranteed approximation ratio of 2 and a polynomial time complexity. The heuristic approach involves considering a specific scenario, called the Mean-Scenario, and running a single instance of MCLP to obtain a solution. This approach is a 2-approximation algorithm, which is used in this thesis, and the research by Coco et al. [4]. In the Mean-Scenario, the benefit of each customer is defined as the average of the upper and lower bounds of the benefits interval, and the corresponding solution is considered. The paper suggests that the problem solved in this research has potential applications in various fields, such as scheduling, resource allocation, and facility location. Finally, the paper provides an informative overview of the proposed algorithm and its potential applications.

This research is crucial to us as we adopt the proposed Mean-Scenario algorithm approach in our thesis. It serves as the foundation of the Mean-Scenario algorithm outlined in Coco et al.'s research [4], which we plan to improve upon by utilizing an approximate algorithm called AAF in this thesis.

Aissi et al. [2] conducted a review paper that delves into the concept of the Min-Max Regret. This paper explores how uncertainty and imprecision in the parameters of combinatorial optimization problems can be structured through scenarios. To address the issue of parameter variations, the article introduces and explains the use of Min-Max Regret criteria, which are frequently used to obtain solutions that are resilient against such changes. These criteria are beneficial in practical applications where anticipating the worst case is essential. The article thoroughly reviews the literature on the Min-Max Regret versions of combinatorial optimization problems. The study focuses on complexity, approximation, and exact resolution in discrete and interval scenario cases. This comprehensive review aims to provide valuable insights into using these criteria for solving practical problems, such as the sensor placement problem in water distribution networks.

Another paper by Furini et al. [13] is explored to obtain the Min-Max Regret. First, the authors work on generalizing a 0-1 knapsack problem, a variation of the classical knapsack problem where the objective is to select a subset of items with maximum profit, subject to a capacity constraint, while taking into account uncertainty in the profit of each item. Then, they calculate the Min-Max Regret using a heuristic and an exact algorithm to solve the problem. Their heuristic algorithm is a local search algorithm that iteratively improves a randomly generated solution by making small changes to the set of selected items. Finally, they also evaluate their work through computational experiments.

Another related paper to the MCLP is done by Berman et al. [14]. This paper deals with the Min-Max Regret Gradual Covering Location Problem (MRGCLP) when insufficient information about the demand weights exists. The MRGCLP involves selecting a set of locations on a network to cover demands with minimum regret. The authors present a two-phase approach to solving the MRGCLP. The first phase involves obtaining an estimate of the demand weights using available information, and the second phase uses the estimated demand weights to solve the MRGCLP. First, the estimated demand weights are used to solve the problem by assuming that the estimated weights are the actual weights.

This solution provides an upper bound on regret. Next, scenarios are generated based on the uncertainty in the demand weights. Each scenario represents a possible realization of the demand weights. For each scenario, the problem is solved using the estimated demand weights as input. The solution obtained for each scenario provides a lower bound on the regret. Finally, the decision maker chooses a solution that minimizes the maximum regret over all the scenarios. They also evaluated their approach through computational experiments, and the results show that it effectively finds high-quality solutions to the MRGCLP, focusing on incomplete information.

Regarding the travel time in MCLP, a recent paper by Chauhan et al. [15] explores drone deliveries instead of usual road transportation, which has applications in emergencies. To improve decision-making, the authors work on a robust multi-period maximum coverage facility location problem considering coverage reliability, or MP-R. First, they use chance constraints to provide a probabilistic constraint satisfaction guarantee to the problem. Then, they develop their final model by integrating the chance-constrained approach with robust optimization. Finally, the paper proposes to use polyhedral uncertainty sets to address the issue of uncertainty in the problem. These sets represent a range of possible values for the variables in the mathematical model. Their experiments show that the proposed model can handle the uncertainty in the problem parameters and provides more robust solutions compared to the baseline models.

The paper by Baldomero-Naranjo et al. [16] explores the Maximal Covering Location Problem. In this paper, the authors consider two types of demand functions, constant and linear, along the network's edges to cover the uncertain demand of their problem. They also call the Min-Max Regret to solve the problem. The job is to place one facility along the network.

Another related work to MCLP is done by Vatsa et al. [17]. The authors propose a mathematical model and a solution algorithm based on meta-heuristics to find the optimal locations for service facilities in a capacitated multi-period setting. They focus on service

uncertainty. When there is service uncertainty, the facility may not be able to serve all the demand that comes its way, potentially leading to inadequate coverage. This means that even if the facility can serve a certain amount of demand, it may not be able to do so due to the unpredictability of server availability. They have tested their algorithm through real-world data to evaluate their work.

A paper studying an uncertain version of MCLP is done by Davari et al. [18]. This paper focuses on a variation of the MCLP where the travel times are considered uncertain. The authors try to find the best possible locations for facilities, such as warehouses, to maximize customer coverage and simultaneously minimize travel time between the facilities and customers. The authors use a heuristic algorithm to solve the problem, and the study results indicate that the proposed model and algorithm can effectively solve the MCLP with uncertain travel times.

Based on a column generation method explored by Davari et al. [19], some algorithms are proposed to solve the MCLP on a large scale. The authors focus on presenting efficient algorithms to handle MCL problems with many demand points and facilities. The study results show that the proposed algorithms can effectively solve large-scale MCLP problems, significantly improving computational time compared to existing methods.

Older research regarding Min-Max Regret is done by Daskin et al. [20]. The authors introduce a new concept called α-reliable p-Minimax regret, which extends the classical Min-Max Regret model for facility location. The α-reliable p-Minimax regret model considers the probability of demand points not being served by any facility and the level of regret associated with each unserved demand point. The model aims to minimize the maximum regret of the unserved demand points, subject to a constraint on the probability of unserved demand points. The authors use a linear programming formulation to present a solution approach for the α-reliable p-Minimax regret model. They also provide a sensitivity analysis to study the impact of the model parameters on the solution and conduct a numerical experiment to evaluate the performance of the proposed model. The experiment

results show that the α-reliable p-Minimax regret model provides a more robust solution than the classical Minimax regret model, as it considers the probability of demand points not being served.

To explore uncertainty in the total number of facilities, another research is done by Current et al. [21]. The authors processed a decision analysis approach in dynamic facility location problems, where the total number of facilities is uncertain. They combine different decision analysis techniques to evaluate the different scenarios which might happen and to make informed decisions about the number and location of facilities. The approach considers each potential scenario's expected costs and benefits, considering the uncertainty about the total number of facilities.

The paper explored by Church et al. [22] presents a novel approach to reserve selection in conservation biology in MCLP. The proposed framework provides a systematic and efficient method for selecting a minimum number of reserves that cover a maximum number of species, considering species' spatial distribution and the cost of protection. The case study results show that the proposed framework can be used to identify a set of reserves that covers many species while minimizing the cost of protection.

Chapter 3

Algorithms for MCLP with Min-Max Regret

Three methods are considered in this chapter to address MCLP under uncertainty. The first one is an exact algorithm based on integer programming. The second is an approximate algorithm based on randomized rounding, and the third is a 2-approximation algorithm using a median scenario [3]. Although, the exact algorithm and the Mean-Scenario algorithm are known algorithms, our objective is to investigate the efficacy of the fractional relaxation of the exact algorithm in generating high-quality inequalities that can enhance the performance of the exact model. Specifically, we aim to include a subset of optimal scenario cost inequalities in the relaxation to achieve this improvement.

3.1 An Exact Algorithm Based on Integer Programming

The exact algorithm solves the problem by being divided into a master problem and a sub-problem [23]. The exact algorithm uses the property that a worst-case scenario for a robust solution can be easily obtained, but modeling the optimal solution of the worst-case scenario requires an exponential number of constraints. The idea behind using the sub-problem is to only generate a small but useful set of inequalities. Firstly, the algorithm solves the master problem to obtain a solution Z. Based on the worst-case scenario obtained from solution Z, the sub-problem is then solved to obtain w, the optimal solution under that scenario. The solution w is then used to update the master problem by getting a constraint. This process continues iteratively until the solution v of the master problem is within a

certain tolerance of the optimal solution. The advantage of this approach is that it solves the MCLP more efficiently, since it uses a bi-level approach, compared to the algorithms which only have one level of formulation. In other words, instead of solving the main problem with an exponential number of constraints, the approach divides the problem into two minor problems [24]. This approach is beneficial when working with a large-scale optimization problem that is difficult to solve directly. The master problem and the sub-problem are solved using the Gurobi optimizer, an optimization software package.

The worst-case scenario is determined by considering the maximum benefit values for customers who are not covered by the robust solution and the minimum benefit values for those who are covered. To see this, consider Figure 3.1, where three facilities have a coverage radius, and the diamonds represent the customers. Given this solution to the robust problem, specific customers are left uncovered while others are covered within the coverage radius. The objective is to identify a worst-case scenario that maximizes the regret of this solution. If the optimal solution of the worst-case scenario covers a customer not currently covered by the robust solution, then the scenario should use the maximum possible value of the benefit for such a customer in order to maximize the regret. If a customer is not covered by the robust solution nor by the optimal solution of the worst-case scenario, then its benefit value does not contribute to the regret and any value can be assigned to the benefit in the worst-case scenario. Therefore, we can assign the maximum value. For the customers who are already covered by the robust solution, if the optimal solution of the worst-case scenario does not cover them, we want the value of their benefit to be chosen as small as possible to obtain the maximum regret. However, suppose the optimal solution also covers the already covered customers. In that case, the choice of benefit values for them does not matter since it will cancel out in the calculation of the regret, therefore, the minimum benefit value is also suitable for the worst-case scenario.

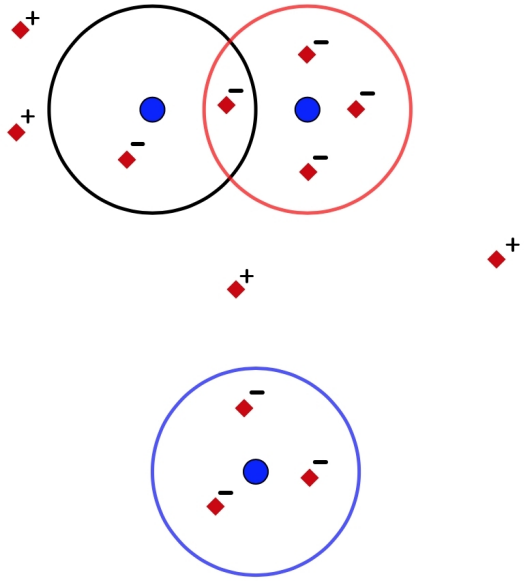

Figure 3.1: Worst-case scenario

In the Integer Programming formulation, the worst-case scenario can be defined by the following expession[3]. The worst-case scenario is indicated by b_i^t.

$$b_i^t = b_i^+ + (b_i^- - b_i^+)z_i \tag{3.1a}$$

If the current solution z_i is 0, b_i^+ will be returned, and if z_i is 1, b_i^- will be returned.

The following is the master problem of the algorithm.

$$\textbf{min} \quad v - \sum_{i \in \mathrm{I}} b_i^- z_i \tag{3.2a}$$

$$\textbf{subject to} \quad z_i \leq \sum_{j \in \mathrm{J}} a_{ij} x_j \quad \forall i \in I \tag{3.2b}$$

$$\sum_{j \in \mathrm{J}} x_j \leq T \tag{3.2c}$$

$$v \geq \sum_{i} ((b_i^+ + (b_i^- - b_i^+)) z_i) w_i^k \quad \forall k \in K \tag{3.2d}$$

$$x_j, z_i \in \{0, 1\} \tag{3.2e}$$

Variable v models the optimal cost of the worst-case scenario represented by the family of equations (3.2d). Also, Let K represent the exhaustive set of all possible 0-1 values indicating all of the ways in which customers can be covered by all possible choices of T facilities. As discussed before, the variables x_j and z_i are the binary decision variables, in which x_j indicates if facility j is placed or not, and z_i indicates if customer i is covered or not. In the objective function of the master problem (3.2a), the maximum regret is minimized by subtracting v, which is the optimal solution under the worst-case scenario, and $b_i^- z_i$, which is the cost of the current solution z_i under the worst-case scenario. The objective function is inspired by [3]. The family of constraints (3.2b) indicate that z_i can cover customer i only if a facility exists in the range of customer i. Also, constraint (3.2c) indicates that the total number of facilities is limited to a constant T. As it is stated that v is the optimal solution under the worst-case scenario, it must be greater than any other possible solution. Therefore, the constraint (3.2d) is included in the master problem. Comparing all feasible solutions to v is infeasible from a computational standpoint. To overcome this challenge, Formulation 3.3 proposes a sub-problem that restricts the possible solutions. This approach will be discussed in the following section.

When the master problem is solved for the first time with some instances, it will give us the initial value of v and a solution of Z, which is a vector representing the values of z

for all the customers from the set I. Based on this solution, the worst-case scenario will be obtained, as discussed earlier. However, variable v contains only a relaxation of the optimal cost of the worst-case scenario. This relaxation is tightened by the sub-problem, which will identify a better value for v by identifying a new vector k of values for w_i^k.

The following is the sub-problem formulation.

$$\textbf{max} \sum_{i \in I} b_i w_i \tag{3.3a}$$

$$\textbf{subject to} \quad w_i \leq \sum_{j \in \mathrm{J}} a_{ij} y_j \tag{3.3b}$$

$$\sum_{j \in \mathrm{J}} y_j \leq T \tag{3.3c}$$

$$y_j, w_i \in \{0,1\} \tag{3.3d}$$

According to the master problem, we have the following: the variable v, the solution Z, and the worst-case scenario based on the solution Z, according to (3.1a). The sub-problem tries to obtain the optimal solution for the worst-case scenario which is then used in the the master problem. The variable w_i gives us a better result than the one already obtained, which is the value of v. Each time, a constraint (3.2d) is added to the master problem. This process repeatedly continues until the following condition does not apply anymore. It is guaranteed that v is the optimal solution. The following is the stop condition for the sub-problem.

$$\sum_{i \in I} (b_i^+ + (b_i^- - b_i^+) z_i) w_i^k \leq v \qquad \forall k \in K \tag{3.4a}$$

We illustrate the relationship between the master problem and the sub-problem with an example based on the Figure 3.1, and the formulation provided in the previous section, the

master problem 3.2, and the sub-problem 3.3.

This example aims to calculate the Min-Max Regret based on Table 2.1, for the three customers A, B, and C on a line, according to the Figure 2.1, with the following intervals of benefits, as discussed before.

Benefit for customer A = [5, 10]

Benefit for customer B = [4, 8]

Benefit for customer C = [3, 20]

Accordingly, the following scenarios are considered.

S_1 = [5, 4, 3], S_2 = [5, 4, 20],

S_3 = [5, 8, 3], S_4 = [5, 8, 20],

S_5 = [10, 4, 3], S_6 = [10, 4, 20],

S_7 = [10, 8, 3], S_8 = [10, 8, 20].

The notation s^t indicates the worst-case scenario in each iteration, in each step of the following approach. Also, the numbers near each variable show the number of the iteration.

When the master problem runs for the first time, it give us the values of $v^1 = 0$ and $Z^1 = (1,1,0)$. The worst-case scenario is obtained according to (3.1a), giving us the following scenario for the solution Z^1. $s^{t^1} = (5,4,20)$, which is the worst-case scenario. According to this scenario, the sub-problem 3.3 will be called to give us a better solution compared to v^1. The answer for the sub-problem is $w^1 = (0,0,1)$. If we compute the new optimal solution, based on s^{t^1} and w^1, it will be 20, which is bigger than $v^1 = 0$, so a new constraint according to the constraint family (3.2d) will be added to the master problem, and the master problem runs with considering this constraint in the second iteration.

In the second iteration, the master problem will give us the new values of $Z^2 = (0,0,1)$ and $v^2 = 3$, and $s^{t^2} = (10,8,3)$. Also, the sub-problem comes up with $w^2 = (1,1,0)$, with the optimal answer of 18, which is greater than 3. Consequently, a new constraint will be

added to the master problem.

For the third and the last test, the master problem will give us the value of $v^3 = 20$, $Z^3 = (1,1,0)$, and $s^{t^3} = (5,4,20)$. The sub-problem will give us $w^3 = (0,0,1)$, and the optimal answer for that is 20, equal to the value of v^3. So, the stop condition applies, and the search will be terminated. Also, the final answer for the Min-Max Regret, which is the value of the objective function of the master problem 3.2, will be 11 and $Z^3 = (1,1,0)$, which is the last solution, considers x_2 as the selected candidate location to allocate the single facility, T.

3.2 A Sub-optimal Algorithm Using Randomized Rounding

Aside from the exact algorithm, an approximate approach to solve the MCLP fractionally rather than an integer is also utilized in this thesis. This method generates the family constraints more rapidly than the exact algorithm, enabling a comparison of results between the approximate algorithm based on randomized rounding *AAF* and those of the Mean-Scenario algorithm. The formulation for this approach is the same as the exact algorithm, with the primary difference being that variables x and z are defined to take on values between 0 and 1 instead of solely 0 or 1. This relaxation computes a lower bound on the optimal robust solution, based on the reserach by van den Berg et al. [25], page 7. The outcome of this method is a fractional solution vector x, which saves the index of the open facilities. To identify a set of T facilities to open, we need to round the fractional values.

The rounding function aims to modify the fractional solution x to values 0 or 1. The process involves setting the initial values of x in an array called $\bar{x}$ to 1 and 0 based on a parameter called c. Then, using randomized rounding, a loop selects T facilities from the remaining elements of x. The final output is an array of 0s and 1s representing the selected facilities.

This function has three inputs: fractional x of size m, T, which is the number of facilities, and the parameter c. The parameter c is a user specified parameter between 0 and 1,

which is compared to the initial values of x. Indeed, the parameter c is a lower bound to determine the values of initial x which are more important to us, and set those to 1, in the first step. The function takes the initial vector of x and if any value in x is greater than or equal to c, the corresponding element sets to 1 in the array $\bar{x}$, and the element's value in x is set to 0. Then, T decreases by 1, and if it becomes negative, the process stops since we can not place more than T facilities. If not, the function loops T times and calculates the sum s of the remaining elements in x.

Next, the function generates a random number RN between 0 and s. It then calculates the prefix sum of x, which is the sum of the first i elements of x. The function then checks if RN is less than the prefix[i]. If it is, the corresponding element in $\bar{x}$ sets to 1, and the element's value in x sets to 0. The loop ends here.

Finally, the function returns the final array $\bar{x}$ of 0 and 1, in which the sum of its values equals T, indicating the number of the chosen facilities.

3.3 A 2-Approximation Algorithm Using Mean-Scenario

The heuristic approach, discussed in the literature review, explored by Adam Kasperski et al. [3], involves considering a particular scenario, known as the mean scenario, and running a single instance of MCLP to obtain a solution. As this method typically yields satisfactory outcomes, it serves as a solid foundation for contrasting the two algorithms derived in the thesis, the Exact Algorithm and the Approximate Algorithm based on Randomized Rounding. More details about this 2- Approximation algorithm is provided by Vazirani et al. [26]. In this algorithm, the benefit of each customer i is defined as $b_i = (b_i^+ + b_i^-)/2$, and the corresponding solution by MCLP is obtained and reported. Accordingly, using this scenario, a solution w is obtained and saved for calculating the regret later. Next, the optimal solution w, which consists of zeros and ones, is used to determine the worst-case scenario, as discussed before. This worst-case scenario is then used to call the MCLP again

and obtain the optimal solution of a new w under the worst-case scenario. The regret can then be computed by subtracting the cost of the worst-case scenario under the new w and the cost of the same scenario under the initial w based on the mean scenario.

Chapter 4

Evaluation of the Algorithms

This thesis uses the exact algorithm based on integer programming, an approximation algorithm based on randomized rounding, and a 2-Aproximation algorithm using a mean scenario to solve the uncertain MCLP with Min-Max Regret. The last algorithm is a heuristic since there are no performance guarantees for it. The exact algorithm optimizes a linear function subject to linear constraints with integer variables. Furthermore, the algorithm divides the problem into a master problem and a sub-problem to solve the robust MCLP efficiently. The approximate algorithm based on randomized rounding fractionally solves the MCLP, and a code in Julia is written to round the fractional values to 0 or 1. Also, the 2-Approximation algorithm uses a mean-scenario to obtain the Min-Max Regret to MCLP. Gurobi optimizer [27] is used to solve all three algorithms under different instances.

4.1 Data Collection and Generation

To evaluate the optimization algorithms of this thesis, the p-median data sets provided by Beasley in the OR-Library [28] are used. It is also noticeable that the data sets are well-known and widely used collections of test instances for various operations research (OR) problems. The library is freely available for academic and research purposes. It can be used as an open-source data set for testing and benchmarking optimization algorithms. These data sets have been used in this thesis. Each data set describes a network, including the number of vertices, edges, and the information on the edge list, including the weight of each edge. Also, a constant T shows the number of facilities to distribute over the network

to cover the customers provided by the data set. The number of vertices varies between 100 and 400, and the number of edges is between 200 and 3200. Also, T has different values between 5 and 100.

In addition to the OR-Library [28], a set of random $m \times m$ matrices, with values 0 and 1, were generated to provide a baseline for comparing the results with the OR-Library instances. More details are provided in the following section.

4.1.1 Generating the Coverage Matrix

A Python code is written to generate the coverage matrix A, with values a_{ij}, which was discussed in the formulation of the algorithm in Chapter 2. Next, a graph G is created from an edge list provided in a text file input.txt. Each line of the file contains two integers representing the nodes connected by an edge and the weight of the edge.

Next, a constant value R, the coverage radius by facility, is defined. Initially, the coverage radius R is chosen to ensure that the density of ones in the matrix falls within the range of two to five percent. This proportion provides a standard base for the coverage matrix [4], and is accomplished by testing various coverage radii to determine which one produces the desired proportion. Subsequently, it increases steadily. The density is defined as the number of ones divided by the total number of elements in the matrix.

To calculate the shortest path between all the nodes in the graph, the code searches over all pairs of nodes using Dijkstra's algorithm, which is used for finding the shortest path between two nodes in a graph. It starts at a node, which is specified by the input, and visits adjacent nodes, updating their shortest distance from the source node. It continues until it reaches the target node or until all reachable nodes have been visited. The algorithm guarantees to find the shortest path if the graph is non-negative weighted and connected, according to the research by Wang et al. [29], on page 1068. If the length of the shortest path is less than R, the value of 1 is set in the matrix. Otherwise, the value of 0 will be considered. The diagonal elements of the matrix are then set to 1 to determine that a node

is always covered by itself.

As previously stated, the second type of instance generated randomly is described as follows. The binary coverage matrices A with a given density were generated randomly. Five different matrices were generated for each size of $m \times m$, which were tested for a set of T values. The density of the matrices is set within the range of 0.02 to 0.05 using the $np.random.uniform$ function in Python. This is a *NumPy* function that generates random numbers from a uniform distribution. The matrices have different sizes between 100×100 to 1000×1000, and the values of the matrices are 0 or 1. The probability of 1 is equal to the density. The generated matrices are set to be symmetric because they try to simulate symmetric distances.

4.1.2 Assigning the Customer's Benefit Values

Two methods have been used to allocate the values of benefit for the customers over the network. Also, to address the uncertainty in the benefit values, each method creates two vectors b_i^- and b_i^+ in the formulation, as discussed before. The number of the values for each array equals the number of customers, known as i.

In the first method, b_i^- is a vector of integers with values between 1 and 10, generated by generating an array of random numbers between 0 and 1 using $np.random.rand()$ function. Then it is multiplied by 9 and added 1 to shift the range from 1 to 10. Next, the resulting values are rounded to the nearest integer using $np.round().astype(int)$ function.

Similarly, b_i^+ is a vector of integers between 15 and 30, generated by generating an array of random numbers between 0 and 1 using $np.random.rand()$ function. Then it is multiplied by 15 and added 15 to shift the range from 15 to 30. Next, the resulting values are rounded to the nearest integer using $np.round(...).astype(int)$ function. These two vectors are the other inputs of the primary optimization problem.

The second method chooses the b_i^- and b_i^+ from an integer interval [1, 100]. Also, a core interval of [40, 60] is considered. Half of the benefit values are picked from the core

interval, and the second half is picked from the main interval. This method will give more variety and larger differences between the values of b_i^- and b_i^+, since using a larger range of values for both creates a greater possibility of generating extreme values of benefits, both high and low.

The first method's output is considered for testing the p-median instances, and the output of the second method is used to test the cases based on the random matrices in the experiments. As a result, there are various values of benefits to solving the MCLP for the two different instances.

4.1.3 Assigning the Number of Facilities, T

The number of facilities assigned to the candidate locations is known as T in the problem of this thesis. This value is a constant, already given as one of the inputs. As discussed earlier, the data sets provided by Beasley [28] also provide the number of the picked facilities to allocate over the network for each problem. Additionally, following a standard approach [4], T is selected as 10, 20, and 30 percent of the total number of customers for the randomly generated instances.

4.1.4 Computation Resources

The experiments of the thesis were run with the Digital Research Alliance of Canada [1]. This is a nationwide establishment that offers high-performance computing (HPC) resources to scholars throughout Canada. Additionally, it involves Cloud Data Repository (CDR), which is a secure platform for data storage and sharing that enables the storage and secure sharing of significant amounts of data. A request was submitted to allocate resources for the jobs through CDR, and the scheduler was asked to give a compute node with 10GB of memory to run the jobs, which are the experiments. A time limit of 3600 seconds is used for running the jobs.

[1] `https://alliancecan.ca`.

4.2 Analysis of Results

All three algorithms were tested using p-median and randomly generated instances. Twenty OR-Lib instances are used with different numbers of vertices for graph G and different coverage radius R. The smallest value of the radius for each p-median instance in the OR-Lib, in the tables, determines the density between two and five percent of ones in the coverage matrix A. As discussed earlier, the matrices for either p-median or random instances are considered square matrices since all the vertices could be potential facility locations. The number of facilities, T, is given for the p-median instances in the data set. For the randomly generated instances, 10, 20, and 30 percent of the number of vertices is chosen for the number of facilities to be allocated.

In the following tables, the exact algorithm is labeled by EA, the approximate algorithm based on randomized rounding is labeled as AAF, and the 2-Approximation algorithm based on the Mean-Scenario is represented by AAM. Each algorithm's results show a ratio of regret, which is the value of the Min-Max Regret divided by the value of the optimal solution v, obtained by MCLP. As the value of v updates at each step, the one that leads to the Min-Max Regret is selected for the calculation of the ratio. This provides a normalized value for the optimal solution for comparisons. It is worth noting that the AAM algorithm does not involve any iterations, which means that there is no specific value of v that can be used to calculate the regret ratio for this algorithm. Consequently, the value of v from the final iteration of AAF is utilized to determine the regret ratio for AAM. The formulation for the regret ratio is defined as follows:

Let v be $\sum_{i\in I}((b_i^+ + (b_i^- - b_i^+))z_i)w_i^k$, in the last iteration, then,

$$\text{regret ratio} = \frac{v - \sum_{i\in I} b_i^- z_i}{v}. \tag{4.1}$$

The equation's numerator indicates the Min-Max Regret for the solution, which measures the difference between the optimal cost for the worst-case scenario and the actual cost of the worst-case scenario. The denominator of the equation is equal to the v value in the

numerator. As previously discussed, *EA* and *AAF* solve the problem iteratively to obtain the Min-Max Regret, based on the generated constraint in each iteration. Consequently, the v value is updated in each iteration until the Min-Max Regret is obtained. Once achieved, the value of v that does not grow anymore is used to calculate the regret ratio in the above formulation. Notably, for the same instance, the value of v utilized in the formulation differs between *AAF* and *EA* since *AAF* solves the MCLP fractionally, while *EA* solves it in binary form.

Tables 4.1 to 4.4 display the outcomes for the OR-Lib instances. Among the experiments conducted, the *EA* failed to produce a solution within the allocated time frame in 20 percent of the instances. In these cases, the *AAF* algorithm had an average regret that was 18 percent higher than the *EA* algorithm in terms of obtaining the optimal solution, but the difference did not exceed 54 percent in 10 percent of the cases. Firstly, this shows the strength of the *AAF* algorithm in solving the instances which were not solved by the *EA* algorithm in the allotted time, and secondly, the small differences between the results of *AAF* and *AAM* indicate the quality of the solutions obtained by *AAF*. Accordingly, the experimental findings reveal that, in terms of both computation time and regret ratio, the performance of *AAF* is comparable to that of *AAM*. Despite *AAM* being faster, as it only deals with a single scenario and does not involve a constraint generation process, *AAF* generates a large number of constraints. However, the experimental results indicate that the difference in the Min-Max Regret between the two algorithms is only nine percent, suggesting that *AAF* can produce solutions with a similar level of optimality to *AAM* while considering a wider range of scenarios.

Also, the results for the randomly generated matrices are presented in tables 4.5 to 4.10. Out of the randomly generated instances, the *EA* algorithm failed to generate a solution within the allotted time frame in 15 percent of the experiments. In such cases, the *AAF* algorithm yielded an average regret that was 10 percent higher than that of the *EA* algorithm in terms of achieving the optimal solution, although the disparity did not surpass 47 percent

in 15 percent of the cases. As anticipated, *AAM* effectively tackles the problem within 25 percent of the regret ratio in *EA* since it only considers a predetermined scenario for the customers' benefit, while *EA* searches for the best scenarios. These observations indicate the importance of the inequalities produced in the *AAF* algorithm known as v constraints. Therefore, I assume that these new constraints bring the value of obtaining sub-optimal solutions to the problem.

Also, as the experiments are conducted on a scheduled cluster, the run time should be regarded as a general indicator of the computational resources since the cluster configuration may differ across experiments.

Some cases are considered as follows for more details about obtaining the optimal solutions by each algorithm. In the p-median 4 in Table 4.1, with $R = 50$, the *EA* approach fails to resolve the problem within the 3600 second time limit. In contrast, *AAF* solves this problem instance, p-median 4 with $R = 50$, in considerably less time, with a ratio of 0.38, while *AAM* achieves a ratio of 0.36 for the same problem. The same case happens for p-median 9 in Table 4.2, with $R = 30$, p-median 13 in Table 4.3, with $R = 20$, p-median 14 in Table 4.3 with $R = 20$, and p-median 18 with $R = 20$, which is considered as an idication of the $AAF's$ sterngth compared to *EA* and *AAM*.

Table 4.1: p-median (1-5)

Instance	**R**	**EA**	**T(s)**	**AAF**	**T(s)**	**AAM**	**T(s)**
P-med1	50	0.5	12.74	0.82	10.5	0.53	10.96
V=100	100	0.22	12.50	0.73	28.39	0.23	10.86
T=5	150	0.00	7.89	0.23	8.56	0.00	10.70
P-med2	50	0.23	598.08	0.56	21.2	0.24	10.96
V=100	100	0.00	10.59	0.33	15.65	0.00	10.92
T=10	150	0.00	13.15	0.00	12.63	0.00	11.03
P-med3	50	0.34	94.20	0.41	23.60	0.35	10.96
V=100	100	0.00	13.21	0.31	15.61	0.00	10.87
T=10	150	0.00	10.41	0.00	15.20	0.00	10.90
P-med4	50	N/A	3600	0.38	23.81	0.36	10.95
V=100	100	0.00	20.09	0.00	20.97	0.00	10.92
T=20	150	0.00	19.80	0.00	20.65	0.00	10.91
	40	0.18	882.03	0.23	22.36	0.19	10.89
P-med5	50	0.00	19.79	0.00	20.67	0.00	10.94
V=100	100	0.00	19.77	0.00	20.75	0.00	10.83
T=33	150	0.00	19.79	0.00	20.76	0.00	10.77

Table 4.2: p-median (6-10)

Instance	R	EA	T(s)	AAF	T(s)	AAM	T(s)
	30	0.56	80.53	0.68	33.77	0.72	11.35
P-med6	50	0.31	167.77	0.52	259.57	0.37	16.27
V=200	100	0.00	9.78	0.05	10.23	0.00	11.35
T=5	150	0.00	9.90	0.00	11.16	0.00	11.48
	30	0.37	470.19	0.54	39.81	0.44	11.35
P-med7	50	0.18	269.58	0.64	76.90	0.20	15.19
V=200	100	0.00	9.36	0.01	9.95	0.00	11.32
T=10	150	0.00	9.81	0.00	10.19	0.00	11.22
	30	0.26	1400.87	0.40	30.36	0.31	10.60
P-med8	50	0.02	81.15	0.18	46.58	0.02	14.38
V=200	100	0.00	10.45	0.00	10.96	0.00	11.44
T=20	150	0.00	10.36	0.00	10.95	0.00	10.62
	30	N/A	3600	0.43	66.44	0.23	10.27
P-med9	50	0.00	10.06	0.06	10.66	0.00	10.43
V=200	100	0.00	10.25	0.00	10.80	0.00	10.45
T=40	150	0.00	10.23	0.00	10.77	0.00	10.45
	25	0.00	9.57	0.00	9.85	0.00	12.46
P-med10	50	0.00	10.83	0.00	10.46	0.00	10.40
V=200	100	0.00	10.62	0.00	11.34	0.00	10.27
T=40	150	0.00	10.99	0.00	10.53	0.00	10.42

Table 4.3: p-median (11-15)

Instance	R	EA	T(s)	AAF	T(s)	AAM	T(s)
	20	0.50	235.62	0.52	52.45	0.61	12.85
P-med11	50	0.02	57.79	0.06	60.61	0.02	24.61
V=200	100	0.00	9.80	0.01	10.93	0.00	12.91
T=5	150	0.00	9.96	0.00	10.49	0.00	12.88
	20	0.47	2985.29	0.67	160.15	0.59	12.78
P-med12	50	0.02	24.61	0.05	29.62	0.02	29.62
V=200	100	0.00	12.92	0.00	13.68	0.00	12.79
T=10	150	0.00	12.88	0.00	13.42	0.00	13.03
	20	N/A	3600	0.60	513.55	0.35	12.80
P-med13	50	0.00	9.98	0.01	10.47	0.00	13.16
V=200	100	0.00	9.60	0.00	8.71	0.00	12.84
T=20	150	0.00	9.58	0.00	10.26	0.00	13.02
	20	N/A	3600	0.38	274.66	0.24	12.84
P-med14	50	0.00	9.68	0.00	10.07	0.00	12.95
V=200	100	0.00	9.51	0.00	10.35	0.00	12.90
T=40	150	0.00	9.55	0.00	9.67	0.00	12.85
	20	0.00	10.20	0.00	10.57	0.00	12.69
P-med15	50	0.00	10.65	0.00	10.95	0.00	13.05
V=200	100	0.00	10.45	0.00	11.29	0.00	12.90
T=40	150	0.00	10.21	0.00	10.44	0.00	12.76

Table 4.4: p-median (16-20)

Instance	R	EA	T(s)	AAF	T(s)	AAM	T(s)
	20	0.16	21.17	0.25	43.68	0.16	25.65
P-med16	50	0.00	11.27	0.10	11.72	0.00	13.65
V=200	100	0.00	10.97	0.00	11.62	0.00	13.27
T=5	150	0.00	10.84	0.00	11.40	0.00	17.36
	20	0.26	117.55	0.33	68.28	0.29	26.62
P-med17	50	0.00	10.40	0.00	10.10	0.00	14.45
V=200	100	0.00	7.83	0.00	11.53	0.00	13.27
T=10	150	0.00	8.86	0.00	9.14	0.00	13.14
	20	N/A	3600	0.41	2244.42	0.21	13.26
P-med18	50	0.00	11.36	0.00	11.33	0.00	14.46
V=200	100	0.00	9.49	0.00	10.11	0.00	13.26
T=20	150	0.00	9.45	0.00	10.00	0.00	13.29
	20	0.00	15.58	0.08	17.08	0.00	12.98
P-med19	50	0.00	26.60	0.00	26.36	0.00	14.30
V=200	100	0.00	18.42	0.00	20.01	0.00	13.23
T=40	150	0.00	18.06	0.00	19.12	0.00	13.33
	20	0.00	16.97	0.00	18.06	0.00	12.97
P-med20	50	0.00	30.58	0.00	25.24	0.00	14.27
V=200	100	0.00	17.66	0.00	19.08	0.00	13.26
T=40	150	0.00	18.00	0.00	18.74	0.00	13.20

An interesting case for the randomly generated instances is in Table 4.5, where $V = 100$ and $T = 20$. In matrix 5, the problem remains unsolved by *EA*, but using *AAF*, it is resolved in 77.93 seconds with a regret ratio of 0.37. Also, *AAM* provides a solution to the problem in almost 13 seconds with a regret ratio of 0.27. As it is shown, *AAF* obtains the sub-optimal solution in the range of only 10 percent from the *AAM* algorithm, which proves the importance of the generated constraints by *AAF*.

Table 4.5: Random Matrices, V=100

Instance	**T**	**EA**	**T(s)**	**AAF**	**T(s)**	**AAM**	**T(s)**
Matrix 1	10	0.38	794.32	0.38	65.41	0.44	12.21
Matrix 1	20	0.18	132.66	0.35	58.95	0.19	12.26
Matrix 1	30	0.04	45.64	0.23	37.81	0.04	16.2
Matrix 2	10	0.36	259.41	0.36	44.69	0.42	12.01
Matrix 2	20	0.13	168.87	0.26	19.84	0.14	9.76
Matrix 2	30	0.03	15.90	0.03	11.80	0.04	12.04
Matrix 3	10	0.22	24.42	0.35	34.06	0.23	24.42
Matrix 3	20	0.09	106.38	0.24	42.73	0.10	9.62
Matrix 3	30	0.01	11.17	0.10	11.85	0.01	11.23
Matrix 4	10	0.33	350.44	0.65	73.84	0.36	11.41
Matrix 4	20	0.14	120.99	0.22	59.14	0.17	13.29
Matrix 4	30	0.00	10.97	0.2	10.04	0.00	11.33
Matrix 5	10	0.35	386.97	0.54	80.56	0.43	11.23
Matrix 5	20	N/A	3600	0.37	77.93	0.27	12.92
Matrix 5	30	0.01	11.43	0.28	9.54	0.01	11.30

Table 4.6: Random Matrices, V=200

Instance	T	EA	T(s)	AAF	T(s)	AAM	T(s)
Matrix 1	20	0.09	3219.85	0.41	830.61	0.1	222.02
Matrix 1	40	0.00	15.41	0.03	10.70	0.00	11.41
Matrix 1	60	0.00	16.41	0.00	11.04	0.00	11.44
Matrix 2	20	N/A	3600	0.58	3409.37	0.22	86.58
Matrix 2	40	0.00	12.44	0.40	11.05	0.00	11.64
Matrix 2	60	0.00	7.91	0.00	10.94	0.00	11.50
Matrix 3	20	N/A	3600	0.46	2225.61	0.18	107.20
Matrix 3	40	0.00	11.84	0.23	12.02	0.00	11.65
Matrix 3	60	0.00	11.35	0.20	11.06	0.00	11.58
Matrix 4	20	N/A	3600	0.43	429.75	0.32	11.64
Matrix 4	40	N/A	3600	0.39	519.72	0.19	17.18
Matrix 4	60	0.00	9.20	0.20	9.68	0.00	11.18
Matrix 5	20	N/A	3600	0.47	1081.53	0.33	17.03
Matrix 5	40	0.04	165.27	0.50	143.84	0.04	33.52
Matrix 5	60	0.00	10.37	0.00	9.55	0.00	11.59

Table 4.7: Random Matrices, V=300

Instance	**T**	**EA**	**T(s)**	**AAF**	**T(s)**	**AAM**	**T(s)**
Matrix 1	30	0.00	155.18	0.51	107.03	0.00	163.45
Matrix 1	60	0.00	11.11	0.10	12.81	0.00	11.58
Matrix 1	90	0.00	9.38	0.00	10.26	0.00	11.56
Matrix 2	30	0.00	9.96	0.39	10.48	0.00	12.23
Matrix 2	60	0.00	9.71	0.18	14.25	0.00	11.76
Matrix 2	90	0.00	10.11	0.00	12.52	0.00	11.68
Matrix 3	30	0.00	121.47	0.41	75.21	0.00	71.38
Matrix 3	60	0.00	8.56	0.19	15.50	0.00	11.91
Matrix 3	90	0.00	8.45	0.00	14.26	0.00	11.50
Matrix 4	30	0.00	29.87	0.35	23.91	0.00	11.64
Matrix 4	60	0.00	22.56	0.22	14.27	0.00	11.37
Matrix 4	90	0.00	22.45	0.00	13.67	0.00	11.50
Matrix 5	30	0.00	9.79	0.13	10.61	0.00	17.03
Matrix 5	60	0.00	10.25	0.03	15.25	0.00	11.63
Matrix 5	90	0.00	10.18	0.00	12.89	0.00	11.49

Table 4.8: Random Matrices, V=400

Instance	**T**	**EA**	**T(s)**	**AAF**	**T(s)**	**AAM**	**T(s)**
Matrix 1	40	0.00	11.34	0.35	16.56	0.00	11.77
Matrix 1	80	0.00	10.11	0.00	15.28	0.00	11.20
Matrix 1	120	0.00	9.69	0.00	13.18	0.00	11.24
Matrix 2	40	0.00	328.96	0.40	269.93	0.00	172.64
Matrix 2	80	0.00	25.24	0.25	14.79	0.00	11.73
Matrix 2	120	0.00	25.04	0.00	15.85	0.00	10.79
Matrix 3	40	0.00	11.19	0.47	10.87	0.00	71.38
Matrix 3	80	0.00	10.19	0.00	15.07	0.00	13.80
Matrix 3	120	0.00	10.10	0.00	15.16	0.00	13.68
Matrix 4	40	0.00	10.78	0.21	13.71	0.00	14.06
Matrix 4	80	0.00	10.19	0.00	15.82	0.00	13.69
Matrix 4	120	0.00	10.21	0.00	12.62	0.00	13.73
Matrix 5	40	0.00	54.21	0.30	51.18	0.00	31.88
Matrix 5	80	0.00	10.11	0.01	13.46	0.00	13.52
Matrix 5	120	0.00	10.05	0.00	15.69	0.00	13.76

Table 4.9: Random Matrices, V=500

Instance	**T**	**EA**	**T(s)**	**AAF**	**T(s)**	**AAM**	**T(s)**
Matrix 1	50	0.00	11.58	0.32	12.27	0.00	15.02
Matrix 1	100	0.00	10.48	0.01	11.74	0.00	13.99
Matrix 1	150	0.00	10.36	0.00	10.31	0.00	13.96
Matrix 2	50	0.00	57.74	0.37	38.78	0.00	48.02
Matrix 2	100	0.00	13.59	0.00	10.57	0.00	14.04
Matrix 2	150	0.00	14.92	0.00	10.66	0.00	14.00
Matrix 3	50	0.00	53.85	0.36	35.43	0.00	49.03
Matrix 3	100	0.00	10.91	0.01	10.83	0.00	13.95
Matrix 3	150	0.00	10.81	0.00	10.69	0.00	14.01
Matrix 4	50	0.00	82.21	0.43	15.06	0.00	111.71
Matrix 4	100	0.00	10.10	0.03	14.94	0.00	13.98
Matrix 4	150	0.00	11.60	0.00	14.96	0.00	13.76
Matrix 5	50	0.00	14.71	0.32	16.25	0.00	23.58
Matrix 5	100	0.00	9.75	0.00	16.55	0.00	13.87
Matrix 5	150	0.00	11.90	0.00	14.26	0.00	13.98

Table 4.10: Random Matrices, V=1000

Instance	**T**	**EA**	**T(s)**	**AAF**	**T(s)**	**AAM**	**T(s)**
Matrix 1	100	0.00	15.69	0.01	14.27	0.00	15.68
Matrix 1	200	0.00	12.93	0.00	14.59	0.00	15.66
Matrix 1	300	0.00	15.60	0.00	13.30	0.00	15.69
Matrix 2	100	0.00	12.75	0.01	14.77	0.00	15.56
Matrix 2	200	0.00	15.13	0.00	13.22	0.00	15.68
Matrix 2	300	0.00	14.64	0.00	14.95	0.00	15.61
Matrix 3	100	0.00	14.92	0.03	13.81	0.00	15.76
Matrix 3	200	0.00	16.91	0.00	14.41	0.00	15.62
Matrix 3	300	0.00	14.27	0.00	14.81	0.00	15.60
Matrix 4	100	0.00	12.85	0.01	15.49	0.00	15.65
Matrix 4	200	0.00	11.30	0.00	13.83	0.00	15.67
Matrix 4	300	0.00	14.91	0.00	13.59	0.00	15.72
Matrix 5	100	0.00	14.90	0.03	15.35	0.00	15.59
Matrix 5	200	0.00	13.75	0.00	18.45	0.00	15.62
Matrix 5	300	0.00	17.61	0.00	15.68	0.00	15.63

To illustrate how the *EA* method works, two different examples of OR-lib and random problems are studied. By analyzing the following graphs, we can see how these methods behave in each example. The graphs demonstrate how v increases with each iteration to reach the optimal solution for finding the Min-Max Regret. The X-axis shows the number of iterations, and the Y-axis shows the v parameter. Two OR-Lib instances were selected. One with $R = 30$, $V = 200$, and $T = 20$ and another with $R = 20$, $V = 200$, and $T = 5$. For the first one, the problem was solved by *EA* in 115 iterations, and *AAF* took only 73 iterations to solve the same problem, as shown in Figure 4.1. Both approaches in the second example required 32 iterations to solve the problem, but *AFF* outperformed *EA* in terms of time. Figure 4.2 illustrates that *AAF* outperforms *EA* because of the faster constraint generation.

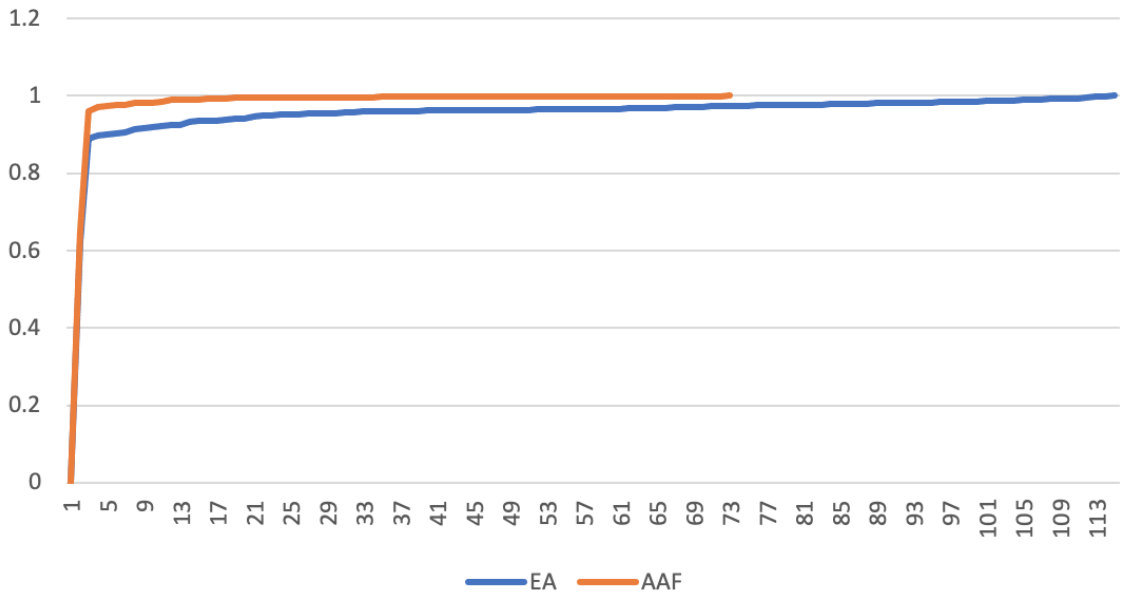

Figure 4.1: p-med8, R=30

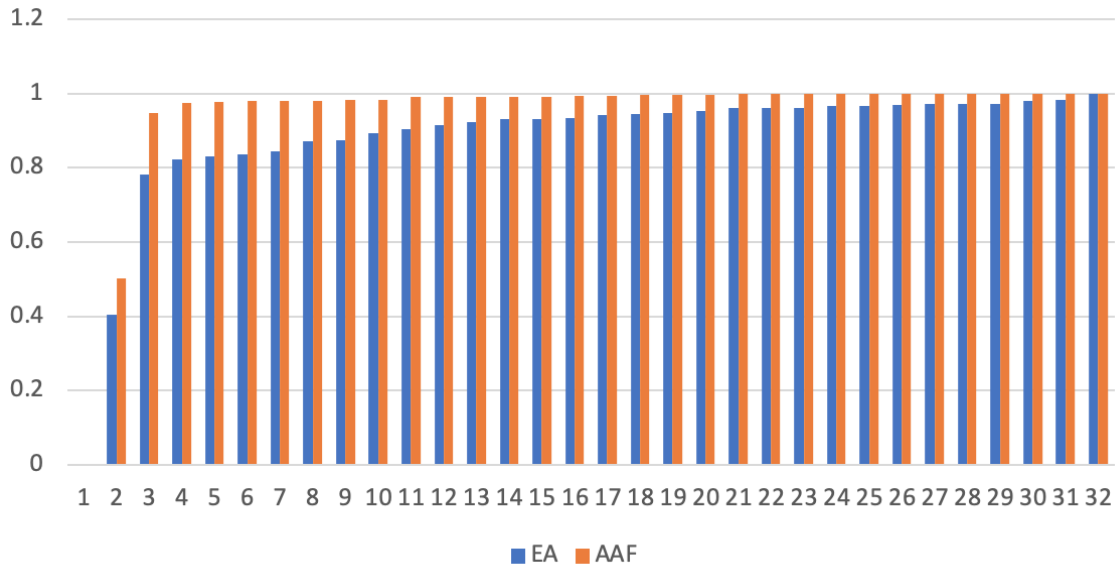

Figure 4.2: p-med11, R=20

The two randomly selected instances are $M1$ with $T = 0.1$ from Table 4.5 and the same instance from Table 4.6. The results show that AAF outperforms EA in terms of time and the number of iterations to find a sub-optimal solution, as illustrated in Figure 4.3. Also, the second randomly selected instance was solved with nearly the same number of iterations by EA and AAF, but AAF solved the problem in almost four times less time than EA. The corresponding graph for this instance can be found in Figure 4.4.

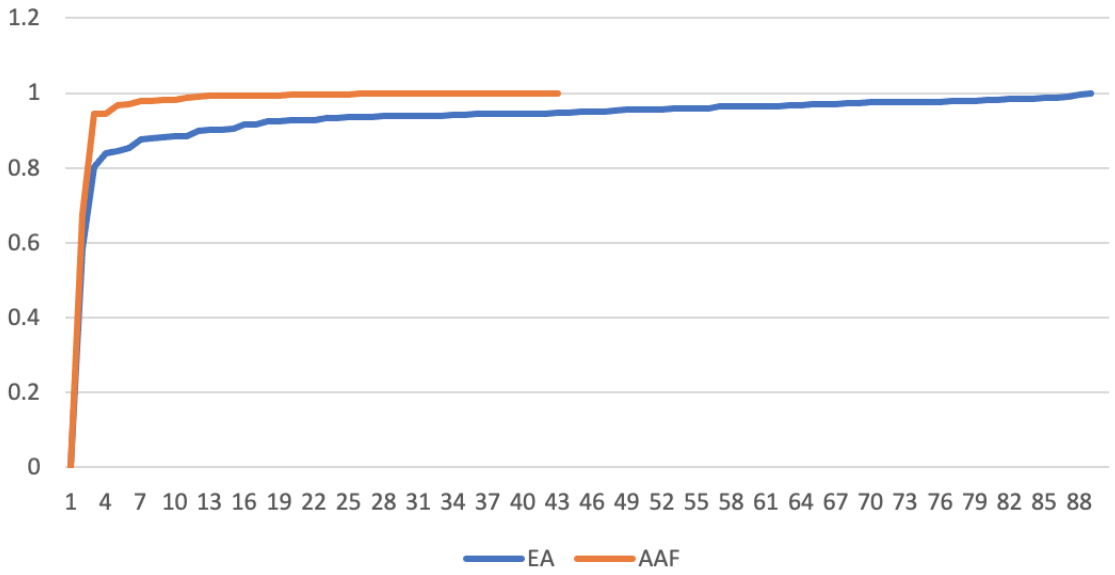

Figure 4.3: Random V=100, M1, T=10

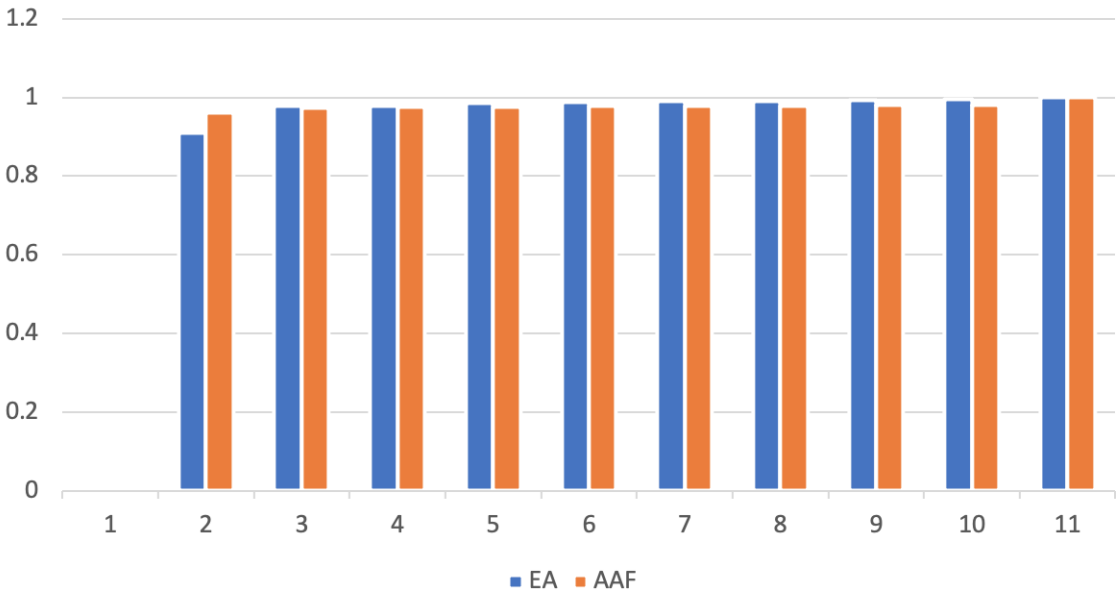

Figure 4.4: Random V=200, M1, T=20

The results presented in the preceding tables and figures show promise in the AAF constraints, to attain the Min-Max Regret objective.

Another solution aspect goes to the variable x and the solution set Z. The variable x indicates the index of the facilities chosen for allocation throughout the network, while Z

denotes the customers covered by the selected facilities. To visualize the representation of customer coverage, the instance p-median 1 with $R = 50$ and $T = 5$ is selected. Figure 4.5 illustrates the facilities set among customers in red, represented as vertices in the network. Figure 4.6 shows the customers covered within the specified radius in green, while the blue ones remain uncovered since the MCLP tries to cover the possible demand with a focus on minimizing the regret, in the problem of this thesis. It should be noted that the OR-Lib instances' locations are unavailable. The process of drawing the graphs is as follows. The algorithm first reads an edge list from a file and creates a graph using the nodes and edges in the list. Next, the positions of the nodes are determined using a layout algorithm provided by Python. The draw function of networks in Python is then called with the graph and node positions as arguments. Additional options such as node size, label size, and font color are also provided to specify how the graph should be drawn. Finally, the graph is plotted.

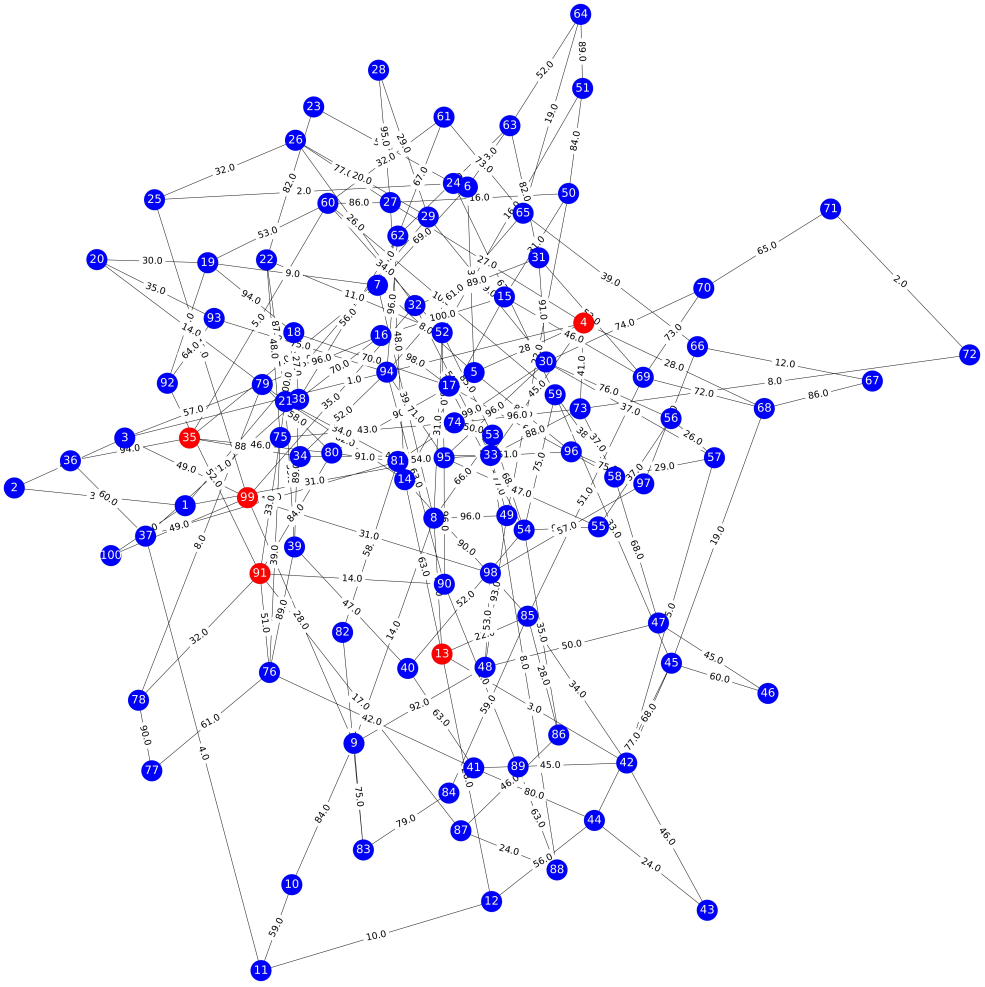

Figure 4.5: p-median 1, R=50, T=5

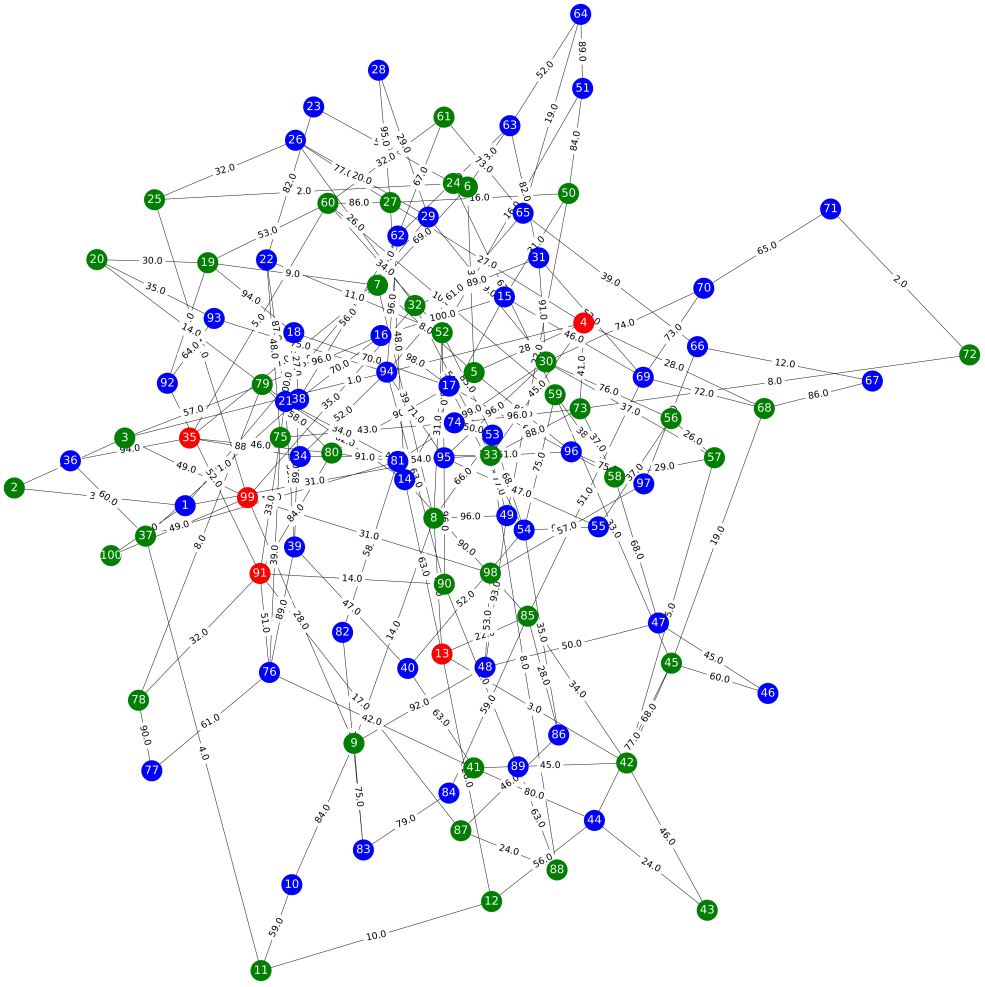

Figure 4.6: p-median 1, R=50, T=5, Covered

4.3 Limitations of the Approaches

In the exact algorithm of this thesis, *EA*, the optimization model's efficacy is contingent on the scale of the problem. It means, as the size of the problem increases, the optimization model may require additional time to be resolved. Since this approach is structured as an integer programming problem [30] [31], finding the optimal solution can be challenging for larger problems. The approximate algorithm AAF reports the result of a randomized rounding procedure whose parameters have not been fine-tuned, and there is potential for improvement in this approach. Also, the 2-Approximation algorithm *AAM* outperforms *EA* and *AAF*, but it does not guarantee optimal solutions to solve the MCLP. While the effectiveness of this approach may be influenced by input parameters like the coverage matrix, benefit value upper and lower bounds, and target variable, it still manages to deliver results within 18 percent of the exact algorithm and nine percent of the 2-Approximation algorithm for the same instances.

Chapter 5

Conclusion and Future Work

The primary focus of this thesis was to address the Maximal Covering Location Problem (MCLP) under uncertain values of customer benefits to obtain the Min-Max Regret. Two types of instances were used to evaluate the proposed approach's performance. Three methods were used to solve these instances, including one exact and two approximate algorithms.

One of the key contributions of this thesis is the generation of novel constraints, which were added to the master problem for *EA* and *AAF*. The results of this study showed that the solutions obtained by *AAF* were within nine percent of the results obtained by *AAM*. Based on this finding, I assume that there is further value in the process of constraint generation by *AAF*. The set of constraints produced by the *AAF* can be further exploited by incorporating the *AAM* algorithm to reinforce the effectiveness of *AAF*. This can also address the open problem posed by Coco et al. [4] aabout further exploration of the 2-Approximation algorithm, based on the mean-scenario.

In more details, the study results showed that the two approximate algorithms could provide solutions when the exact algorithm could not solve a problem within the allotted time. The *EA* algorithm provided a baseline for the optimal solution, while the approximate algorithm *AAF* provided solutions with significantly lower computational costs. This trade-off between computational cost and solution quality is essential in practical applications where the optimization problem may be too large or complex to solve optimally within a reasonable time frame.

Regarding future work, there are several directions to pursue. While randomized rounding in the AAF reports effective results compared to EA and AAM, it may not fully reflect the quality of the constraints generated by the algorithm. Therefore, other approaches to strengthen randomized rounding are suggested. As an example, the parameter c is a critical factor in determining how the rounding is performed. Exploring multiple alternative rounding constants with varying values of c could potentially lead to new insights and solutions for a broader range of problems and improves the quality of the solutions obtained by AAF. These experiments would yield diverse and interesting outcomes and help researchers better understand the trade-offs between different rounding methods and their effects on optimization performance. Also, in the fractional algorithm AFF, the Gurobi optimizer continues to run the master problem until it terminates, producing numerous v constraints. Rather than proceeding with the rounding procedure to obtain rounded solutions for x and z, it would be interesting to investigate what would happen if the master problem were to run again, taking into account the v constraints, with the variables x and z restored to their original binary types. Then, an exact approach can be employed using binary variables to generate new constraints that can be compared to the constraints generated by AAF. This provides an opportunity to compare the two sets of constraints for v to obtain the optimal solution.

Bibliography

[1] Richard Church and Charles ReVelle. The maximal covering location problem. -, 32(1):101–118, 1974.

[2] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European journal of operational research*, 197(2):427–438, 2009.

[3] Adam Kasperski and Paweł Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177–180, 2006.

[4] Amadeu A Coco, Andréa Cynthia Santos, and Thiago F Noronha. Formulation and algorithms for the robust maximal covering location problem. *Electronic Notes in Discrete Mathematics*, 64:145–154, 2018.

[5] Alan T Murray. Maximal coverage location problem: impacts, significance, and evolution. *International Regional Science Review*, 39(1):5–27, 2016.

[6] Oded Berman and Dmitry Krass. The generalized maximal covering location problem. *Computers & Operations Research*, 29(6):563–581, 2002.

[7] Robert G Cromley, Jie Lin, and David A Merwin. Evaluating representation and scale error in the maximal covering location problem using gis and intelligent areal interpolation. *International Journal of Geographical Information Science*, 26(3):495–517, 2012.

[8] Roberto Diéguez Galvão and Charles ReVelle. A lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, 88(1):114–123, 1996.

[9] Igor Averbakh and Oded Berman. Minmax regret median location on a network under uncertainty. *INFORMS Journal on Computing*, 12(2):104–110, 2000.

[10] Soheil Davari, Mohammad Hossein Fazel Zarandi, and I Burhan Turksen. A greedy variable neighborhood search heuristic for the maximal covering location problem with fuzzy coverage radii. *Knowledge-Based Systems*, 41:68–76, 2013.

[11] Lawrence V Snyder. Facility location under uncertainty: a review. *IIE transactions*, 38(7):547–564, 2006.

[12] Amadeu A Coco, Andréa Cynthia Santos, and Thiago F Noronha. Robust min-max regret covering problems. *Computational Optimization and Applications*, 83(1):111–141, 2022.

[13] Fabio Furini, Manuel Iori, Silvano Martello, and Mutsunori Yagiura. Heuristic and exact algorithms for the interval min–max regret knapsack problem. *INFORMS Journal on Computing*, 27(2):392–405, 2015.

[14] Oded Berman and Jiamin Wang. The minmax regret gradual covering location problem on a network with incomplete information of demand weights. *European Journal of Operational Research*, 208(3):233–238, 2011.

[15] Darshan Rajesh Chauhan, Avinash Unnikrishnan, Miguel A Figliozzi, and Stephen D Boyles. Robust multi-period maximum coverage drone facility location problem considering coverage reliability. *Transportation Research Record: Journal of the Transportation Research Board*, 2022.

[16] Marta Baldomero-Naranjo, Jörg Kalcsics, and Antonio M Rodríguez-Chía. Minmax regret maximal covering location problems with edge demands. *Computers & Operations Research*, 130:105181, 2021.

[17] Amit Kumar Vatsa and Sachin Jayaswal. Capacitated multi-period maximal covering location problem with server uncertainty. *European Journal of Operational Research*, 289(3):1107–1126, 2021.

[18] Soheil Davari, Mohammad Hossein Fazel Zarandi, and Ahmad Hemmati. Maximal covering location problem (mclp) with fuzzy travel times. *Expert Systems with Applications*, 38(12):14535–14541, 2011.

[19] MH Fazel Zarandi, Soheil Davari, and SA Haddad Sisakht. The large scale maximal covering location problem. *Scientia Iranica*, 18(6):1564–1570, 2011.

[20] Mark S Daskin, Susan M Hesse, and Charles S Revelle. α-reliable p-minimax regret: A new model for strategic facility location modeling. *Location science*, 5(4):227–246, 1997.

[21] John Current, Samuel Ratick, and Charles ReVelle. Dynamic facility location when the total number of facilities is uncertain: A decision analysis approach. *European journal of operational research*, 110(3):597–609, 1998.

[22] Richard L Church, David M Stoms, and Frank W Davis. Reserve selection as a maximal covering location problem. *Biological conservation*, 76(2):105–112, 1996.

[23] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10:237–260, 1972.

[24] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[25] PL Van den Berg, GJ Kommer, and B Zuzáková. Linear formulation for the maximum expected coverage location model with fractional coverage. *Operations Research for Health Care*, 8:33–41, 2016.

[26] Kamal Jain and Vijay V Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM (JACM)*, 48(2):274–296, 2001.

[27] Edward Rothberg. Gurobi optimization: three founders strive to build a different kind of optimization software company. *OR/MS Today*, 41(2):22–25, 2014.

[28] John E Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

[29] Huijuan Wang, Yuan Yu, and Quanbo Yuan. Application of dijkstra algorithm in robot path-planning. In *2011 second international conference on mechanic automation and control engineering*, pages 1067–1069. IEEE, 2011.

[30] Gerhard J Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 185–207. Springer, 2003.

[31] William J Welch. Algorithmic complexity: three np-hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15(1):17–25, 1982.