Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico-Matemáticas

Bilevel Facility Location Problems: Theory and Applications

Una tesis presentada por:

Martha Selene Casas Ramírez

Como requisito parcial para obtener el grado de:

Doctor en Ciencias

con

Orientación en Matemáticas

San Nicolás de los Garza, Nuevo León, México
Julio 2017

Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico-Matemáticas

Bilevel Facility Location Problems: Theory and Applications

Una tesis presentada por:

Martha Selene Casas Ramírez

Como requisito parcial para obtener el grado de:

Doctor en Ciencias

con

Orientación en Matemáticas

San Nicolás de los Garza, Nuevo León, México
Julio 2017

Autonomous University of Nuevo Leon

Faculty of Physical-Mathematical Sciences

Center for Research in Physical-Mathematical Sciences

The committee members, hereby, certify that have read the thesis presented by Martha Selene Casas Ramírez and that it is fully adequate in scope and quality as a partial requirement for the degree of Doctor in Science with Major in Mathematics.

Dr. José Fernando Camacho Vallejo
Principal Advisor

Dr. Dolores Edwiges Luna Reyes
Co-advisor

Dr. Juan Antonio Díaz García
Committee Member

Dr. Omar Jorge Ibarra Rojas
Committee Member

Dr. Álvaro Eduardo Cordero Franco
Committee Member

Vo. Bo.

Dr. José Fernando Camacho Vallejo
Coordinator of the Graduate Programs in Science with Major in Mathematics

San Nicolás de los Garza, Nuevo León, México
July 2017

"For me, there is no emotion comparable to the one produced by creative science, whether it is in science or in art, literature or other occupations of the human intellect. My message, directed primarily at youth, is that if they are inclined to science, follow it, because it will not fail to be provided unequaled satisfactions. It is true that the moments of discouragement and frustration abound, but these are soon forgotten, while satisfactions are never forgotten."

Severo Ochoa

Acknowledgements

This thesis, although it required effort and dedication from me and the thesis director, would not have been possible to complete without the disinterested cooperation of each and every one of the people who have been a good support in good and bad times. Thank you for giving me your support and knowledge.

I thank God for accompanying me and guiding me throughout my life, for being my strength in moments of weakness and for giving me a life full of learning, experiences, health, and happiness. For always putting the right people in my way that incite me to become a better person every day. I infinitely thank you for giving me back much more than I had asked for.

I am especially grateful to my academic father, Dr. Fernando Camacho, for having trusted me and agreeing to carry out this thesis under his direction. For giving me the opportunity to draw on your professional and personal capacity and experience. Thank you for your generosity of sharing your knowledge with me, your patience and your ability to guide not only the development of this thesis but also my professional training. For your comments, suggestions, advice, and criticism that are reflected in this work. Also, for presenting me increasingly difficult challenges and for supporting me to realize them. For motivating me to continue in times of frustration and despair. Thank you for the support of your projects, for the procedures that you managed for me in the faculty and in the postgraduate program. For forming me professionally. It was a privilege for me to have been his thesis student for these 5 years (master's and doctorate). It is impossible to describe in a few lines the degree of gratitude I feel, as well as admiration for his teaching and research.

I am grateful to the members of the committee, Dr. Dolores Luna, Dr. Juan Díaz, Dr. Omar Ibarra and Dr. Eduardo Cordero for their time invested, for their keen observations and valuable suggestions in the construction and improvement of this work. I would like to extend my gratitude to Dr. Iris Martínez for having accepted a collaboration, for having dedicated time and effort to a part of this work.

I want to thank Dr. Carmen Galé, Dr. Herminia Calvete, Dr. Dolores Luna and Dr. Juan Díaz for having received me at the research stays. Thank you for your time invested, for all your knowledge imparted and for your constant feedback. For making my research stays were pleasant.

I thank my family, my mother Margarita Ramírez, my sister Marlene Casas and my nephew Christopher Rodríguez, for always supporting me unconditionally. For your advice, for sharing my sorrows and joys, for always giving me encouragement and for teaching me never to surrender. For motivating me when I felt that I could no longer continue, for having unconditionally understood my absences and my bad moments. Thank you for everything, I love you.

I want to thank all those people who motivated me and supported me at various points in this process: Yajaira Rodríguez, Jesús Rodríguez, Olga Hernández, Alejandra Cerda, Oscar Ovalle, Mayra Pardo, Marcia Cruz, Sarina Córdoba, María Jesús Castán, María Ángeles Villalba, Pilar Moix, Diana Huerta, Claudia Morales, Luisa Martínez, Ana Hernández, Diana Sánchez, Dámaris Dávila, Mario Aguirre, Lilian López, Ivonne Valdés, Isela Hernández, Pamela Palomo, Pedro Loera and the boys of the University Pastoral (Zaragoza).

I especially thank Jorge Garza and Adriana Arias for always giving me words of encouragement when I needed them and for being with me in the good and bad moments. For all your understanding, for forgiving my absences and for always supporting my decisions. Thank you for your unconditional friendship.

I would like to thank the Autonomous University of Nuevo Leon (UANL), the Faculty (Department) of Physical-Mathematical Sciences (FCFM) and the Center for Research in Physical-Mathematical Sciences (CICFIM) for all the support provided during my postgraduate course. Furthermore, it is my pleasure to extend my gratitude to the the Mexican National Council for Science and Technology (CONACYT) for the supporting scholarship and the mixed scholarships for my research stays held at the University of Zaragoza (Spain) and the University of the Americas Puebla (Mexico).

Contents

Chapter 1

Introduction

In this doctoral thesis we focus on studying facility location problems considering customer preferences. In these problems, there is a set of customers or users who demand a service or product that must be supplied by one or more facilities. By facilities it is understood some object or structure that offers some service to customers. One of the most important assumptions is that customers have established their own preferences over the facilities and should be taken into account in the customer-facility assignment. In real life, customers choose facilities based on costs, preferences, a predetermined contract, or a loyalty coefficient, among others. That is, they are free to choose the facilities that will serve them.

The situation described above is commonly modeled by bilevel programming, where the upper level corresponds to location decisions to optimize a predefined criteria, such as, minimize location and distribution costs or maximize the demand covered by the facilities; and the lower level is associated to -customer allocation- to optimize customer preferences. The hierarchy among both levels is justified because the decision taken in the upper level directly affects the decision's space in the lower level.

In this thesis, different variants of bilevel facility location problems considering customer preferences are studied:

- The uncapacitated facility location problem (named UFLBP) is analyzed. To find feasible solutions for the problem we propose a hybrid metaheuristic method that combines evolutionary algorithms and path relinking. In addition, we perform a study that shows the importance of solving to optimality the lower level problem.
- A variant of the p-median problem (named p-median BPO) is studied. We develop two reformulations of the mathematical bilevel program and also, we propose an equivalent single-level reformulation of the problem. In addition, we propose a scatter search algorithm to find good quality feasible solutions.
- A maximal covering location problem (named MCLP) is also studied. We introduce a mathematical model, develop two classic reformulations and propose a genetic algorithm to find good quality feasible solutions. Also, we propose a reformulation that reduces the problem into a single-level one. In addition, we propose a GRASP-Tabu

hybrid algorithm which obtains good quality solutions in a reasonable computational time for larger instances.

- A variant of the capacitated facility location problem with unitary demands (named CFLBP) is investigated. We analyze and exploit the properties of the problem, which allow us to obtain feasible bilevel solutions. We propose a genetic algorithm to find good quality feasible solutions for the problem.
- A second variant of the CFLBP with customer demands (named BCFLP) is considered. Customer demands add a significant degree of complexity to the problem, because the lower level is NP-hard. Therefore, there is no guarantee of finding the optimal solution of the lower level, and hence, to the bilevel problem. We introduce the necessary definitions to obtain an approximated inducible region. For the bilevel model, we propose three versions of a heuristic algorithm based on cross entropy. The obtained results provide important information on the impact of not being able to obtain bilevel feasible solutions.

All location problems in this thesis take into account the preferences of the customers. The lower level problem is associated to the customers' allocation. To have a well defined bilevel programming problem, the lower level problem must have a unique optimal solution for any fixed upper level's decision. For all uncapacitated problems, we guarantee the uniqueness of the optimum of the lower level, representing customer preferences as consecutive integer numbers. The proof of this results is given in [136]. In these problems, the lower level problem is solved to optimality by using an ordered matrix of preferences, which is a methodology proposed in [92]. When capacities and demands are considered in the lower level, the optimistic version is considered to have well posed the bilevel problem. In other words, if multiple optimal solutions for a given upper level's decision can be found, then the one which incurs in the best upper level's objective function value is selected. This occurs in CFLBP and BCFLP.

1.1 Location problems

Location theory is one of the areas of operational research that has stood out most because of its real-life applications. In [47] facility location problems are defined as "given a certain metric space and a set of known points, determine a series of additional points to optimize a function of the distance between new and existing points". Theory of facility location and some applications can be found in [43]. There are several ways to classify location problems; for instance, based on the type of constraints and objective function.

One of the classical location problems is the p-median problem. The objective of this problem is to locate p facilities in such a way that distribution costs are minimized. The formulation of the problem, its properties and some contributions about it can be found in [47].

Another problem that has been extensively studied in the literature is the uncapacitated facility location problem. In this problem, the number of facilities to locate is not fixed beforehand. Instead, fixed costs are considered for each potential facility location. This problem has also been studied considering capacity constraints, which adds a degree of difficulty. In this situation, facilities have a capacity that should not be exceeded and the demand points or customers have a demand that must be supplied. For more information, see [47].

Coverage problems have also attracted the attention of researchers. The main purpose of these types of problems is to locate facilities to provide a service to customers that are within a predetermined threshold distance (see [34]).

There are other types of location problems that are not mentioned above, for example: sequential location, undesirable or obnoxious facilities, fixed-charge, p-center, anti-covering, hub location, competitive location, facility location under uncertainty, etc (see [47], [101], and [43]). In some cases, for these problems the notion of preferences may be considered.

The fundamental difference between the classic problems mentioned above and those studied in this thesis is that, customers are allocated based on their preferences and not to the nearest facility, which is the common criterion for allocation.

1.2 Bilevel programming

Multilevel programming is a mathematical modelling technique used to consider nested components involved in a particular situation. It is an important tool for handling situations where the decision process can not be assumed to be performed in a simultaneuos way, because the decision made by one of the decision makers directly affects the decision of the other decision makers. In [25] was first defined the area of multilevel programming as a generalization of mathematical programming. The particular case in which only two decision levels are considered corresponds to Bilevel Programming (BP). In other words, instead of considering two interrelated processes in an independent way, BP is convenient for simultaneously considering both parties in a hierarchized manner during the decision-making processes.

In a general way, a bilevel programming problem can be approached from two perspectives: game theory and mathematical programming. From the game theory point of view, there are two competing decision-makers associated with the upper level and lower level problems (leader and follower, respectively). In that hierarchized sequential game, first the leader establishes its strategy, then the follower rationally reacts by choosing its best strategy having complete knowledge of the leader's decision. Also, it is assumed that perfect information is given since the leader will decide its strategy first, and, after that, the follower will decide his own strategy.

From the mathematical programming point of view, a bilevel programming problem consists of a mathematical programming problem in which the variables can be partitioned

into two subsets. The main issue is that one of the subsets of variables must be determined by the optimal solution of another mathematical programming problem; that is, a bilevel programming problem is a mathematical programming problem with another optimization problem in its constraints. To clarify the idea, the general formulation presented in [11] is shown below:

$$\min_{x \in X} \quad F(x, y) \tag{1.1}$$

$$\text{subject to:} \quad G(x, y) \leq 0 \tag{1.2}$$

$$\min_{y \in Y} \quad f(x, y) \tag{1.3}$$

$$\text{subject to:} \quad g(x, y) \leq 0 \tag{1.4}$$

where $F, f : R^n \times R^m \to R^l$, $G : R^n \times R^m \to R^p$, and $g : R^n \times R^m \to R^q$.

In this formulation, it can be seen that the decision variables of the leader is the vector x and the vector y is defined by the follower. The leader aims to minimize the function $F(x, y)$ under the constraints (1.2) and (1.3). The follower wants to optimize the function $f(x, y)$ considering the constraints (1.4).

In problems modelled with bilevel programming, even when the functions are continuous and bounded, there can be no guarantee that the optimal solution exists. Most of these problems are non-convex, and if there are multiple leaders or multiple followers, we must first find the Nash equilibrium between them. The bilevel programming problems are NP-hard (see [8]). Some properties, solution algorithms, optimality conditions and computational difficulties can be found in [11], [38], [140], [9], [39], [13], [10], [13] and [37].

Interesting literature reviews regarding applications in planning, distribution, location, pricing, network design, humanitarian logistics, and environmental models, modelled as bilevel programming problems, appear in [11], [31], [71], [137], [71] and [5].

1.3 Motivation and contribution

Although facility location problems have been of interest to the scientific community, there is only one paper where the p-median problem with customer preferences modelled with bilevel programming is studied. In this thesis and in [3], some reformulations are proposed to reduce the bilevel problem into a single-level one. The difference between both works is that we propose a heuristic algorithm to solve the p-median problem and they propose another heuristic but to solve the reformulation of the problem.

We also propose a hybrid algorithm for the UFLBP. This algorithm outperforms the evolutionary algorithm in the literature. Besides, we show the impact in the algorithm's performance when solving the lower level through three different exact or heuristic approaches.

To the best of our knowledge, the capacitated version of the problem has not been studied, may be because the lower level problem is NP-hard. The incorporation of capacity constraints in the facilities and demand associated to each customer, converts the lower level problem into the well-known generalized assignment problem (GAP). Due to the complexity of the GAP, the follower's optimal solution cannot be computed in a straightforward manner. We propose appropriate definitions of the concepts for approximating the inducible region for handling solutions that do not belong to the bilevel feasible region. Also, we explored the case when demands are unitary (CFLBP). By considering this assumption, we exploit structural properties of the lower level problem. The resulting problem is a transportation one, therefore the problem can be solved to optimality. As the capacity and demand are integers, then there is at least one integer optimal solution.

We also propose the first model in the literature that handles a covering problem considering customer preferences (MCLP). The problem considers firms that are already in the market and a new firm that wants to locate p facilities in order to maximize the captured demand. In the same manner than the other problems studied in this thesis, customers are allowed to freely choose their allocation to the facilities within a threshold distance. In the upper level, facilities are located in order to maximize the demand covered, whereas in the lower level, customers are allocated to the facilities based on their preferences to maximize a utility function.

1.4 Objective

The primary objective of this thesis is to study facility location problems with customer preferences and to propose solution methodologies that provide good quality solutions at a low computational cost.

For each problem studied, we analyze the properties of the model and develop classical reformulations of bilevel programming to reduce the problem into a single-level. Due to the computational time required to solve the reformulations, we propose heuristic algorithms that provide near optimal solutions with reasonable computational time.

1.5 Structure of the thesis

Based on the problems mentioned above, the chapters of this thesis are divided according to the classification of the facility location problems. In the Chapter 2 a literature review is presented. In Chapter 3, the uncapacitated facility location problem (UFLBP) with customer preferences is presented. A hybrid algorithm is developed for solving a set of benchmark instances. The algorithm hybridizes an evolutionary algorithm with path relinking. In Chapter 4, the p-median bilevel problem with order (p-median BPO) is presented. To solve the problem, an algorithm based on scatter search is developed. In Chapter 5, the bilevel maximal covering location problem (MCLP) is proposed. The problem is solved via reformulations

and by a genetic algorithm. Then, to obtain better results, we propose a hybrid algorithm that combines GRASP an tabu search.

In Chapter 6, the bilevel capacitated facility location problem with customer patronization towards a list of preferences is introduced. We analyze two cases for the problem. In the first version, the lower level corresponds to a transportation problem (CFLBP), and a genetic algorithm is proposed to solve it. Finally, we considered a general case, where the lower level is the well-know generalized assignment problem (BCFLP). Three versions of a cross entropy method are proposed to solve this problem, in two of them semi-feasible solutions are obtained. In Chapter 7, the conclusions of this thesis are presented. Finally, an appendix with the detailed information of each article derived from this study is included. Also, the research visits, participations in conferences, summer/winter schools and some distinctions are mentioned.

Chapter 2

Literature Review

Location theory is an area of operations research which has attracted the attention of researchers due to the necessity to locate facilities that provide a service or product requested by users. One of the main problems is named the facility location problem, which stems from the Weber's problem. The latter consists in determining the point that minimizes the sum of the Euclidean distances from that point to all other given points (see [139]). In the facility location problem there is a set of customers that are distributed in a predefined space. Customers desire that their demands of a particular service or product are met by one or more facilities. The problem is to determine where to locate facilities and how customers would be allocated in order to minimize location and distribution costs associated with that particular decision.

There are different versions of facility location problems without capacity constraints. For example, the simple plant location problem (SPLP) arises when the facility has an infinite capacity and can meet customer demands, the model for this problem was proposed in [78]. Another version of the problem appears when a new objective function is considered as in [107]. The authors include the most popular objectives functions of location models and penalize the distance between the customer and the facility according to the position occupied by the customer. A taxonomy of location models including relevant issues and a classification is provided in [34].

An important consideration is that in classical facility location problems, customers are allocated by the locator. But, it could appear a situation in which the locator is not in charge of this allocation. Instead of that, customers are free to select the facility that will serve them. For instance, practical applications can be found in public sectors in which customers behavior cannot be imposed by the locator. This situation can be studied considering customers as other decision makers with a predefined hierarchy among the locator and them. One way to achieve this is to use customer preferences. In other words, each customer establishes an order for ranking the most preferred facilities based on its own criterion.

Facility location problems considering customer preferences have been studied with a bilevel programming (BP) approach. For example, in the uncapacitated bilevel facility location problem. A leader seeks to open facilities aiming to minimize the sum of locating and distributing costs, while a follower allocates customers to their most preferred facility. The

follower's decision will affect the distributing cost, modifying the leader's objective function. The customers' freedom to choose the facilities that will serve them adds more reality to the problem. In real life, customers choose facilities based on costs, preferences, a predetermined contract, or a loyalty coefficient, among others. Next, some papers devoted to location problems considering customer preferences are discussed.

Integer single-level formulations for the SPLP had been proposed in the literature, where customer preferences are taken into account by including additional constraints. In fact, [66] is the first paper in which a customer's ordered list of preferences (assuming an unknown number of facilities should be located -the simple plant location problem with order (SPLPO)) was considered. A greedy heuristic based on branch and bound was developed to solve the SPLPO. Numerical experiments on small-size instances (5 facilities and 8 customers) were conducted to validate the proposed solution method. Later, [8] demonstrated that the facility location problem with customer preferences and its variations are NP-hard problems. In [26], new valid inequalities yielding tightened bounds for the integer problem are proposed. The valid inequalities were added to the formulation which was then solved by commercial solver, a reduction in the integrality gap is obtained in reasonable computational time.

Although the uncapacitated facility location problem with customer preferences can be modeled as a single-level problem, an alternative and natural manner for formulating it is as a bilevel programming problem. [67] is the seminal paper in which a bilevel model was developed for this problem. Since bilevel programming problems are non-convex and very difficult to solve, a single-level reformulation was made. Also, the authors used pseudo-boolean functions for a relaxed version of the problem obtaining valid lower bounds. In [136], the bilevel problem was reduced to a single-level one by using clique inequalities; this technique allows the existence of a new family of valid inequalities instead of increasing the number of variables. The authors found a lower bound which is not worse than the one obtained in [67].

The SPLPO is also studied in [135], in which some valid inequalities related with customer preferences are introduced. A reduction of the bilevel problem into a single-level one is made. Then, a linear relaxation of the single-level resulting model is made yielding lower bounds. A simulated annealing method was implemented for obtaining upper bounds. Finally, considering both bounds, a branch and cut technique was applied for solving the reduced problem. In [92], a comparison between three metaheuristics is presented. The authors directly solved the bilevel version of the SPLPO. The implemented metaheuristics are based on a swarm particle optimization, simulated annealing and an adapted neighborhood search method. For dealing with the optimal solution of the lower level problem, they rearranged the customer preferences matrix. By doing this, they reduced the required time for solving the follower's problem with an optimizer. Computational testing was conducted for large-size instances showing that the performance of the adapted neighborhood search method outperforms the other algorithms.

Also, in [22], the bilevel version of the SPLPO is considered and two reformulations using the primal-dual relationships of the lower level problem are proposed. As a result of the reformulation, a single-level mixed integer programming problem arises. After showing

the limitations for optimally solving both reformulations with an optimizer, an evolutionary algorithm that considers Stackelberg's equilibrium was proposed. The evolutionary algorithm was applied to solve the bilevel version of the problem, not the reformulations; it could solve instances up to 500 facilities and 1000 customers showing good performance. Also, the algorithm reached the optimal solutions in more than half of the runs and, if not possible, small optimality gaps were found.

In the case that the total number of facilities to be located is known and no fixed cost associated with the facilities are considered, the problem is called p-median (see [64]). Several surveys have been published summarizing and differentiating interesting contributions related with this problem (see [131], [97], [91], [116], [54], [49]). As it has been mentioned in [47], due to its nature, the p-median problem could be applied for modelling a wide range of applications. For example, for locating hospitals, schools or warehouses (geographical decisions). Also, for clustering, assigning tasks or scheduling events (non-geographical). From another perspective, it has applications in public decision making since the objective of the p-median problem could be seen as maximizing accessibility to the facilities. From the private decision making point of view, the problem can be used in location decisions where distribution costs need to be minimized. In [73] is showed that this problem is NP-hard. Efficient exact methods and heuristics have been developed. For a detailed review about both kind of solution methods, see [112] and [98], respectively.

After intensive research, we found only two articles that consider customer preferences in p-median problems. First, in [3] presented a bilevel formulation and described some single-level reformulations yielding to lower bounds. The reformulations are based on different versions of pseudo-booleans functions. Then, a ranking among the obtained lower bounds is done. Moreover, a hybrid genetic algorithm with local search was designed for obtaining upper bounds. Their algorithm considers four crossover types (uniform, greedy, single-point, and path relinking) and local search procedures as mutations (Lin-Kernighan heuristic). They validated the algorithm using instances involving 100 facilities and 100 customers. Their results showed that solving the reformulations is time consuming, and the optimum value is not obtained, whereas the hybrid genetic algorithm finds better solutions in less time. The work in [3] differs from the work presented in Chapter 4 of this thesis mainly in terms of the fact that their heuristic is applied to the single-level reformulation of the problem. Additionally, our proposed methodology only explores on the location decision space while the allocation of the customers is optimally done, and it is capable to obtain good quality solutions for larger size instances in reasonable computational time.

Later, in [21], a similar bilevel model than the one studied in this thesis is presented. The main difference is that in addition to the cardinality constraint, fixed costs for locating facilities were considered. This fact differentiates that problem with the variant of the p-median problem under study in this thesis. Note that in the case when homogeneous fixed costs are assumed, both models are the same. However, in the instances tested in [21] heterogeneous fixed costs were considered. For solving that problem, two single-level reformulations were developed. Numerical experimentations were performed over a set of instances concluding that exact methods require significant computational effort. Therefore, this thesis formally

presents the p-median problem with order and proposes an efficient hybrid algorithm to solve its bilevel model.

Other problems that have a natural modelling with bilevel programming are the competitive facility location (CFL) problems. In [70], the concept of competition between two firms for locating facilities is introduced. The problem was treated as a two stage process; in the first stage, both firms decide the location of the facilities, while in the second stage the firms choose the optimal prices for the offered products. CFL problems can be divided in two main categories (see [65]): the $(r|X_p)$-medianoid and the $(r|p)$-centroid problems. Both problems are modelled under the Stackelberg or leader-follower approach, which results in bilevel models. In the $(r|X_p)$-medianoid, the follower locates r facilities assuming that the leader has p facilities from a set of potential sites X_p. In the $(r|p)$-centroid problem the leader locates p facilities knowing that the follower will react locating r facilities. This kind of problems can also be seen as sequential discrete models (see [52]). Spatial market competition is introduced, differentiating this problem from others. The assumption, that the spatial market is a closed linear segment in which the customers demands are uniformly distributed among the segment, is made. Also, inelastic demands, homogeneous products being offered at facilities, and both firms considering mill pricing, are expected. Therefore, customers will buy the cheaper products without regard to the distance from the facilities, and both firms will have the same cost function. As a result of that paper, the Hotelling's law was established, which states that both firms will cluster their facilities to the center of the market. Besides the discussion about the Stackelberg equilibrium, an analysis for some scenarios in which the Nash equilibrium does not exist is presented. Finally, different variations of the problem are mentioned, such as the optimization of a social function, and the existence of product transportation in multidimensional spaces, among others.

After Hotellings' contribution, many researchers were motivated to tackle these problems. In [46], a very complete taxonomy that involves many specific components of the competitive facility location problem is given. The proposed taxonomy contains the number of involved decision-makers, the price policy, the characteristics of the game (sequential, simultaneous, etc), and the customer's behavior. For a detailed review of competitive facility location problems, the reader is referred to [45], [76], [104] and [122].

Under the bilevel discrete competitive location problem framework, but considering preferences for allocating customers to facilities, we can mention [14]. In that study, two competing companies which successively will locate their facilities to maximize their profits were considered. Beresnev [14] identifies the following three components in the problem: 1) a leader company who seeks to locate their facilities considering the follower's response, 2) a follower company who considers the facilities located by the leader and makes its decision to maximize its own market capture, and 3) a set of customers who are free to choose the facility that will meet their demand. The selected rule for allocating customers to facilities is based on a pre-established list of preferences that customers have regarding being supplied by the facilities. To solve the problem, upper bounds are computed through an auxiliary pseudo-boolean function. Later, in [15], an extension of the previous work is presented. Cooperative and non-cooperative solutions for the same problem are presented. Also, the previous upper

bounds are used to obtain initial solutions, which are improved by two local search algorithms. Numerical results show that better performance occurs when a generalized neighborhood is considered in the local search.

None of the papers above mentioned considers capacity constraints. After an intensive literature search regarding this issue, only [27] was found. In that paper, a problem in which the leader locates the facilities and decides their corresponding capacity for minimizing the costs is considered. The follower selects the fraction of customer demands satisfied by each facility aiming to maximize the profit given from the customers-facilities allocation. The follower's decision affects a set of constraints in the leader's problems, complicating the bilevel problem. A decomposition approach for solving the problem is proposed showing good performance. The main difference with the problem proposed in Chapter 5 is that in [27] the lower level problem is a linear programming problem, which can be optimally solved by exact methods.

One of the problems addressed in the present thesis is the capacitated facility location problem with customer preferences under a bilevel programming scheme. The lack of papers dedicated to this particular problem is possibly due to the fact that the resulting lower level problem is NP-hard. The incorporation of capacity constraints and customer demand constraints converts the lower level problem into the well-known generalized assignment problem (GAP). Due to the complexity of the GAP, the follower's optimal solution cannot be computed in a straightforward manner. Feasible bilevel solutions requires the follower's optimal response for each leader decision -that is, points in the inducible region; a NP-hard problem cannot be optimally solved, however (in general). Also, if a refined method that obtains a good quality solution for the follower is implemented, then the performance (in terms of computational time) of the algorithm designed for solving the bilevel problem will be negatively affected.

In the majority of papers that analyze bilevel programming problems, the incumbent lower level problem can be optimally solved by an optimizer or an exact method. A small number of papers handle a lower level problem which is NP-hard. The common solution methodology is to develop an efficient heuristic algorithm to obtain an acceptable follower's response; by doing this, semi-feasible solutions for the bilevel problem are found. The importance of having tightened bounds for the lower level problem is evident.

To illustrate this, some papers dealing with a complex lower level problem are presented. For example, [89] considers a generalization of bilevel network design with congestion effects. In that problem, the follower's decision variables are given by the solution of an equilibrium problem formulated as a variational inequality. Four heuristic procedures are proposed for solving the problem, providing upper and lower bounds for the optimal bilevel solution. Also, [55] studies a bilevel urban transportation network design model. The upper level concerns a network design problem and the lower level involves a signal setting problem. A Scatter Search algorithm is implemented for solving the bilevel problem. To avoid finding the optimal solution for the follower, a local approach was designed to get the follower's response. Moreover, the signal setting problem was formulated as an asymmetric equilibrium

assignment problem in which the variables related with signal setting and traffic flows are determined by the assignment procedure. Under this scheme, the decision is related only with the topological configuration; by doing this, it is easier to obtain approximate bilevel solutions.

In [19] a new formulation for the ring star problem as a bilevel model is proposed; in such, they considered the existence of a leader and two independent followers. One of the followers must solve a travelling salesman problem and the use of a greedy algorithm with 2-opt and 3-opt local searches is applied. Then, in [23] a topological design of Local Area Networks bilevel problem is considered. The lower level must construct a capacitated spanning tree and it is solved by a greedy constructive algorithm similar than Kruskal's algorithm. Also, in [6] an ant colony optimization algorithm for solving a bilevel production-distribution problem was implemented. They illustrated the behavior of the algorithm when the lower level is solved in an exact manner or by a heuristic procedure. In the case when the heuristic procedure was applied to the lower level, it corresponds to a differential evolution algorithm.

In [125], a mathematical model for the problem in which two competing firms locate the same number of facilities is presented; the authors propose two alternatives for obtaining good solutions. First, they solved the leader problem with a heuristic and the follower problem in an exact way and, then, applied the heuristic algorithm to both levels of the problem. Numerical results show the first approach requires a significant amount of time, while the second approach decreases the computational time, but also worsens the quality of the bilevel solution. Also, [16] deals with a competitive facility location problem, in particular with the (r/X_p)-medianoid. A greedy algorithm was proposed for estimating an approximate follower's solution for each leader's decision. In [77], a competitive facility location problem in which a company operates the existing facilities and a new company enters the market is analyzed. The leader is the company that desires to enter the market and the follower is the company that controls the facilities already existing in the market. In this problem, the follower can react in different ways, that is, by opening new facilities, closing some existing ones, or adjusting the attractiveness levels of the facilities. Both levels are modeled as mixed-integer non-linear problems. A tabu search method is proposed for solving the problem. The follower's solution is found through a branch and bound method applied to the relaxation of the lower level problem, yielding a lower bound for its objective function. If integrality is obtained despite the relaxation, then a bilevel feasible solution is obtained. Other solution approaches are proposed, such as, the ϵ-optimal solution's method which enumerates all the possible locations for the follower, and then, the optimal solution of the resulting non-linear problem is computed. Also, a reformulation is made by using KKT optimality conditions and solved by an algorithm called GMIN-αBB; only small size instances were able to be tested in a reasonable computational time.

The hub location problem is another problem that is related with the facility location problem. Let be $G = (N, A)$ a graph where N is the set of nodes and A is the set of edges. A flow w_{ij} should be sent from each node i to each node j, $(i, j \in N)$. An alternative is to select some nodes to become hubs and use them as redistribution points for achieveing a more efficient flow in the network. The problem consists of deciding which nodes should become hubs and how the flow in the network should be redistributed. When a non-hub node has to be

assigned to exactly one hub, it is a single-allocation hub location problem. In many situations a limit exists on the maximum amount of flow which can be processed in each hub. These capacity constraints lead to the capacitated single-allocation hub location problem (CSA-HLP). In single-allocation models, binary variables are required in the allocation phase, multiple allocation allows different delivery patterns which implies the use of continuous variables and simplifies the problem. For more information see [32], [4] and [24].

In [108], an alternative formulation based on the ordered median objective function, which unifies several hub location models -namely the Single Allocation Ordered Median Hub Location problem (SA-OMHLP), is introduced. This problem provides a common framework to represent many of the considered criteria in the literature of hub location. In this model, single-allocation means that all the outgoing flow is delivered through the same hub, but the incoming flow can come from different hubs. Recently, [109] deals with the capacitated version of this problem. These authors recognize that although capacitated models are more realistic, the difficulty to solve them increases with respect to their uncapacitated versions. They used the most promising formulations of the non-capacitated version of the problem (the "covering" formulations) and tried to adapted for the capacitated problem. The model incorporates capacity constraints on the incoming flow at the hubs coming from origin sites or even simpler, on the number of non-hub nodes assigned to each hub.

Having an ordering problem within a location one increases its complexity in both, the formulation and the methodology of solution. The Discrete Ordered Median Problem (DOMP) was introduced in [100], in which two formulations are proposed: as an integer linear program and as an integer nonlinear program. Then, in [18] an alternative integer linear programming formulation for the DOMP is proposed. A comparison with the existing ones is made showing that the proposed formulation is better. Moreover, some properties regarding optimal solutions that allows the elimination of a subset of variables are found. Taking advance of the properties, a branch and bound algorithm was developed and used for solving a set of benchmark instances.

In [72], an extension of the DOMP called the ordered capacitated facility location problem is proposed and it is modelled from three different points of view. In the first model the demands can be split; in the second model fixed costs for locating facilities are considered; and in the third model the shipping and locating costs are taken into account in the objective function. Also, in the third model they examine three approaches for incorporating shipping costs: i) customers pay this cost, ii) distribution centers pay this cost, and iii) logistic providers pay this cost. To consider the above mentioned approaches they used the objective function proposed in [101]. The locating and shipping costs are ordered before the evaluation of the objective function is made.

A hierarchical location model for public facility planning is presented in [133]. Taking a p-median model as a basis, a new model that adds constraints on the minimum and maximum capacity of facilities and deletes the constraint on the number of open facilities is derived. The authors analyze the spatial pattern of customer-facility assignments resulting from different assignment constraints. Location models with capacity constraints do not have the single and

closest assignment properties. Facilities have a limited capacity and customers are diverted to other facilities, or customers are "captured" to ensure the minimum capacity of a facility. In public location models, a solution that makes that customers belonging to the same facility are split or users from neighboring sites will be assigned to different facilities, are difficult to interpret by decision makers and difficult to explain to customers.

Another classical facility location problem is the covering problem. Covering problems have also attracted researchers' attention for the past 50 years due to their significant impact in real-life situations. The main purpose of these problems is to locate facilities that can provide services to customers within a predetermined coverage radius. This is a reasonable assumption specially in those cases where some type of service will be useless above a certain distance. Classical covering problems, were the coverage radius is given, can be divided into set covering and maximal covering. The set covering problem was introduced in [64], where the aim was to minimize the number of facilities required to cover all demand points. The maximal covering location problem (MCLP) was proposed in [29], where a limited number of facilities has to be located to maximize the demand covered. Since then, many extensions have been proposed and many applications in different domains have been studied. Detailed surveys on covering location and its applications in real-life situations can be found in [124], [49] and [57].

After an intensive literature review, only in [82] customer preferences were considered for covering problems. The problem maximizes the weighted covered demand with respect to customer preferences. They also considered a minimum and maximum number of customers served by a facility. A method based on Lagrangian relaxation and primal heuristics was proposed to obtain upper and lower bounds for the optimal solutions. The proposed algorithm provided good quality solutions.

In [82], there is a decision maker who controls both variables (allocation and location) and he/she gives an integrated solution taking into account customer preferences. However, if customers can decide their allocation among the open facilities, then the problem is approached from a quite different point of view. In the latter situation, there are two decision makers involved in the decision process and a hierarchy exists among them. Bilevel programming is very suitable for handling this situation. It is assumed, that upper level will decide the location of the facilities and that customers will be allocated in the lower level. This is the approach considered in this thesis.

Chapter 3

Uncapacitated Facility Location Problems

3.1 Analyzing the performance of a hybrid heuristic for a bilevel location problem using different approaches to tackle the lower level

To obtain bilevel feasible solutions, the lower level problem must be optimally solved by an optimizer or by an exact method. But, this is not always possible and the current problem needs to be solved somehow. Hence, a heuristic procedure that balances efficiency and computational effort would be desired. Commonly, the use of heuristic procedures for solving bilevel problems results very costly in terms of computational time due to the number of times that the lower level problem is solved. Then, the exploration of methodologies in which the search is guided without the resolution of the lower level turns out to be a matter of interest, see [6], [127], [128] and [129].

The objective of this work is to show the impact of not solving the lower level in an optimal manner during the resolution of a bilevel programming problem. We select a problem where the lower level problem is not difficult to be optimally solved. The motivation of this is for comparing the performance of the case when solving to optimality and when not. We propose a hybrid algorithm for solving the bilevel version of the uncapacitated facility location problem with customer preferences. Also, a discussion about the effects that result from solving the allocation problem with an optimizer, with an alternative exact procedure or with a methodology that avoids solving it at each step is presented. The article is organized as follows: in section 3.1.1 we present the classical mathematical bilevel formulation of the problem. Section 3.1.2 describes the proposed algorithm that hybridizes an evolutionary algorithm with path relinking. The computational experimentation and the results are shown in section 3.1.3.

3.1.1 Description of the problem

In this section the bilevel problem and its mathematical formulation are described. The bilevel model considered in this paper is proposed in [136], where the leader aims to minimize locating and distributing costs. The follower aims to minimize the customer preferences. The sets, parameters, and decision variables involved in the mathematical formulation and the assumptions considered during the research are presented next.

Let $i \in I$ and $j \in J$ be the indices of the potential facilities and customers, respectively. Let c_{ij} represent the costs of supplying all demand of customer j from a facility located at i. Let f_i denote the cost of locating facility i. Finally, g_{ij} represents customer j preference towards facility i.

The sets of variables are:

$$y_i = \begin{cases} 1, & \text{if a facility } i \text{ is located} \\ 0, & \text{otherwise} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1, & \text{if facility } i \text{ satisfies the demand of customer } j \\ 0, & \text{otherwise} \end{cases}$$

The following two assumptions are considered in the problem:

- Customer preferences are represented by an ordered list, where 1 indicates the most preferred facitity and $|I|$-th represents the least preferred facility.
- Uncapacitated facilities, therefore, a facility can supply multiple customers but a customer must be served by a single facility.

The mathematical model for the UFLBP is as follows:

$$\min_{y,x} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{3.1}$$

$$\text{subject to: } y_i \in \{0,1\} \qquad \forall i \in I \tag{3.2}$$

$$x \in \arg\min \quad \sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} \tag{3.3}$$

$$\text{subject to:} \quad \sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{3.4}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \tag{3.5}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{3.6}$$

The problem in the upper level is defined by (3.1)-(3.3), where (3.1) represents the leader's objective function who seeks to minimize both location and distribution costs, (3.2) establish the binary constraints for each variable y_i and (3.3) is the constraint that indicates

that variables x_{ij} are controlled by the follower, these variables are determined by the optimal solution of the lower level. This problem is defined by equations (3.3)-(3.6); the follower's objective function is defined in (3.3) that minimizes the ordered preferences of the customers, (3.4) ensure that each customer is supplied by a single facility, (3.5) indicates that customers allocation can be made only to the located facilities, and finally, (3.6) indicates the binary constraints for the decision variables x_{ij}.

The existence and uniqueness of the lower level's solution is guaranteed because for this study, customer preferences are different from each other. In other words, it is not allowed to assign the same preference value to more than one facility. The proof which validates the latter is shown in [136]. Under this assumption, it is guaranteed that the bilevel problem is well defined.

3.1.2 A hybrid evolutionary algorithm with path relinking

In this section, the hybrid algorithm and its components are described. Hybrid algorithms combine advantages of two or more heuristics to find good quality solutions for a problem. The combination usually obtains either better solutions or a quicker convergence than the use of only one heuristic. An example of hybrid algorithms can be found in [62]. In that paper, the authors identify connections and contrasts between genetic algorithms and tabu search. In [126], a variable population-size genetic algorithm and a particle swarm optimization algorithm are hybridized, the computational results in benchmark functions showed that the hybrid algorithm has a good performance.

In this chapter a hybridization between an evolutionary algorithm and path relinking is proposed. The main idea for including the path relinking scheme is basically to substitute the classical random crossover. The motivations for developing an evolutionary algorithm (EA) are: a population based algorithm allows the search space to be expanded due to the obtaining of several solutions and not only one; and it is well-known that EAs can handle not-easy-to-solve problems in an efficient manner.

The EA consists in three mechanisms: in the first one, the construction of the initial population containing feasible solutions is made; the second mechanism concerns to the application of the genetic operators -crossover and mutation-; and the last one is the selection mechanism to implement the survival-of-the-fittest principle, which depends of two features, quality and diversity. We decided to implement a combinatorial method called Path Relinking (PR) for reaching a local optimum, this procedure is added into the crossover. Path relinking begins with a pair of good quality solutions, parts from one of them and changes its components, once at a time, to convert this initial solution into the second one. In [134] a brief description of the PR is given, here the authors mentioned that the main objective is to explore the search path between a set of (two) solutions. Additionally, PR generates a set of new solutions from the good solutions in the set. Also in [115], PR is described as an intensification strategy to explore trajectories connecting elite solutions obtained by heuristic methods. The

PR proposed in this study, explores paths using only the leader's solutions. If the second solution is identical to the first one, then the method stops and returns the best solution found in the trajectory. It is important to highlight that since the aim of the proposed algorithm is to solve a bilevel problem, to compute the objective function value, the follower's problem is solved in each movement.

The addition of PR to other algorithmic frameworks has shown good performance and has attracted the attention of researchers. For example, in [115] there is a description of the hybridization of path relinking with Scatter Search (SS) and a Genetic Algorithm (GA), and showed that the use of PR almost always improves the performance of a heuristic. Also, in [111] a project scheduling problem is considered, where a hybridization of PR and a GA is developed to find good quality feasible solutions for the problem. The performance of the hybrid algorithm is tested using a set of instances available, and they showed that hybridization is efficient for this specific problem. Moreover, the combination of PR with SS is analyzed in [40]. They solved the capacitated p-median problem (CpMP). In the CpMP it is required to partition over a set of costumers, each with a given demand, in exactly p clusters. They conducted a series of experiments on different sets of test instances, and the results were satisfactory in terms of the quality of the solutions found. Furthermore, they mention that the combination of PR with SS gave good results. Recently, in [51] and [103] path relinking is also hybridized with other heuristics showing good results. Based on these results, we were motivated for hybridizing this procedure within the evolutionary framework.

The principal components involved in the hybrid algorithm are detailed and depicted in Algorithm 3.1.1. It can be seen that an initial population is generated (P_t) in $GenerateInitial\ Population$ and then the fitness of each solution of it is calculated in $Fitness$. After, the selection of the parents is carried out through tournaments in $Selection$ and enter to the genetic operators: crossover and mutation. This process is repeated until it reaches number of generations, that is, $t = Generations$.

Algorithm 3.1.1 Pseudocode Evolutionary algorithm.

```
1:  procedure EVOLUTIONARY ALGORITHM
2:      P_t ← ∅
3:      P_t ← GenerateInitialPopulation(P_t)
4:      f ← Fitness(P_t)
5:      while t < Generations do
6:          P_{t+1} ← Selection(P_t, P_{t+1}, f)
7:          P_{t+1} ← Crossover(P_t)
8:          P_{t+1} ← Mutation(P_t)
9:          P_t ← P_{t+1}
10:         t ← t + 1
11:     end while
12:     return P_t
13: end procedure
```

Solution encoding: The leader's solutions are represented as a binary strings of size $|I|$, that indicate if the facility is opened (value 1) or not (value 0). The follower's problem must be solved and the solution must be used to compute the leader's objective function value.

Initial population: In this mechanism, a predetermined number of solutions is generated. Feasible solutions are created in a random manner. For each solution, a vector of size $|I|$ of independent random numbers uniformly distributed over the interval [0,1] are generated. That vector if filled out component by component as follows: if the respective number is less (greater) than 0.5, then the corresponding facility remains closed (opened).

Selection mechanism: The strategy used is a tournament selection. This procedure consists in randomly matching each solution with another one from the population, and the best of both solutions is selected.

Crossover (Path relinking): The path relinking procedure is used in the crossover. In this case, pairs of solutions are selected in a random way from the population and a crossover point is randomly selected. Here, the first solution through movements becomes into the second one, but only after the crossover point. That is, both solutions are identical as from the crossover point. Then, in each movement a new solution is generated so a trajectory is created. After, the two best solutions are chosen. The crossover point is used in order have a starting point for changing the elements of the first solution. This means that if the first component in the first solution is different at the second solution after the crossover point, this one will be changed; that is, if component is 1, then it will be changed into 0; or vice versa. This procedure is repeated for all subsequent elements until all components are explored.

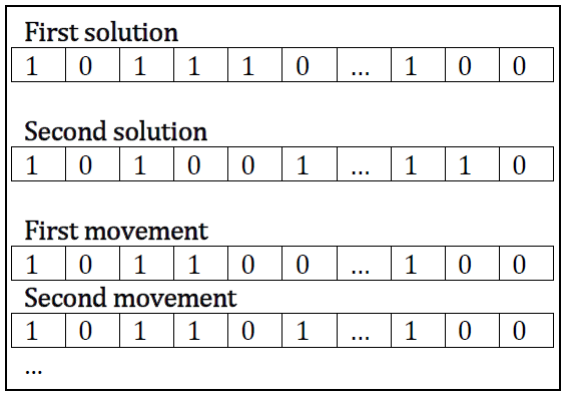
A pseudocode of PR is shown in Algorithm 3.1.2. Here, after having generated the crossing point, the amount of different elements there are after this point in the two solutions is calculated, i.e., D. Then, for each of these components, the change is made from 0 to 1 or vice versa in the solution (y') in $Change$. Then, the lower level is solved in $SolveLowerLevel$ considering this change (x) and the fitness is calculated ($c(y')$) in $Fitness$. If this value is better than the incumbent, the change is made (y_{new}).

Algorithm 3.1.2 Pseudocode of path relinking.

```
1:  procedure PATH RELINKING(y_1, y_2)
2:      y_new ← ∅
3:      D ← d(y_1, y_2)
4:      min ← ∞
5:      for i = 1 : NumDiference do
6:          y' ← Change(y_1, D, i)
7:          x ← SolveLowerLevel(y')
8:          c(y') ← Fitness(y', x)
9:          if c(y') < min then
10:             min ← c(y')
11:             y_new ← y'
12:         end if
13:     end for
14:     return y_new
15: end procedure
```

In Figure (3.1), an illustrative example of PR is showed. It can be seen two initial solutions. Suppose that the crossover point occurred between the fourth and fifth component of the solutions. Note that the fifth component of the first solution is 1, and that of the second solution is 0, then in the first movement the solution generated will be identical to the first solution but the fifth component will have the value of 0. Now, the value of the sixth component of the first solution is 0 and that of the second solution is 1, then that change is made and that generates a second solution in the second movement. The process is repeated until all possible changes are explored, that is, until a solution that is identical before the crossover point to the first parent and identical to the second after the crossover point is generated.

Figure 3.1: Path relinking.



Mutation: The mutation involves changing some components of the string from 0 to 1 or vice versa. The associated probability for switching a component is given by a random number generated for each component. Then, the components of the solutions are mutated in an independent manner; that is, the mutation of a component does not affect the mutation probability of another one.

Finally, population is updated in an elitist way. We now comment on the performance of the proposed algorithm using a set of benchmark instances.

We used different methodologies to solve the UFLBP. First, we considered the evolutionary algorithm (EA) proposed in [22] and hybridized it with a path relinking procedure (PR) included in the crossover phase. Furthermore, within this hybridization, we propose three alternative ways to solve the lower level problem, which are: (i) to optimality by using a commercial solver, (ii) to optimality by using an ordered matrix of preferences proposed in [92] and (iii) by not solving the lower level at each movement explored in the PR.

The latter alternative is based on the fact that for having feasible solutions for the bilevel problem, the lower level needs to be optimally solved for each leader's solution. It is evident that a local procedure, such as PR, a new leader's solution is obtained in each move and for measuring the impact of that specific move the lower level problem needs to be solved. Hence, the computational cost for following that scheme is high for a bilevel problem. Thus, it is interesting to investigate the possibility of not solving the lower level for the new leader's solution during all the PR procedure. By doing this, semi-feasible solutions (feasible for the leader but not optimally for the follower) are being considered in the PR but after the methodology is done, the lower level is optimally solved for the incumbent solution. That is, bilevel feasibility is achieved with the latter.

Owing to the high computational cost of using an optimizer for solving the lower level problem, the methodology proposed in [93] is considered, wherein the customer preferences matrix is rearranged by ordering it into another matrix with ordered facilities indices (based on the customer preferences). The methodology is explained below.

For each customer $j \in J$, the array with potential facilities is ordered according to the preferences of customer j, where the most preferred facility index is at the beginning of the array and the least preferred is at the end. The arrays of all customers are stored in a matrix called the matrix of ordered preferences. Then, Each customer $j \in J$ is allocated to the first open facility according to the ordered preferences specified above. The process is repeated until all customers are allocated to an open facility. Thus, optimal allocation of the customers to the located facilities is obtained.

3.1.3 Computational experimentation

The experimentation was conducted on 3.60 GHz Intel Core i7-4790 with 32.00GB RAM running under Windows 8.1 Pro operative system. The algorithms were implemented on Visual Studio Express 2012 with C++ and for the first alternative, the lower level was solved with CPLEX 12.6.1. The set of benchmark instances is the same that the one used in [22]. The set consists of 36 instances of three different sizes: 12 small, 12 medium, and 12 large-size instances. The small-size ones contain 50 facilities and 50 customers. On the other hand, the medium-size instances contains 50 facilities and 75 customers. Finally, 75 facilities and 100 customers are considered in the large-size instances.

Initial testing indicates that the required time to solve the small-size instances with the proposed hybrid algorithm (EA+PR) with CPLEX is 4410.455 seconds on average. As a consequence of the excessive computational time required by CPLEX for solving the lower level, we decided to discard this approach. Hence, we used the ordered matrix of customer preferences for solving the lower level.

In the case where the lower level was not optimally solved during each move in the PR (this approach will be referred as EA+PRw), two approaches for allocating customers were proposed. For example, consider the situation presented in Figure 3.2.a, where an ordered pair associated with each customer represents the nearest open facility and the most preferred open facility, respectively. Also, consider the case when facility [3] is opened and facility [4] is closed, under the approach when the lower level is not solved at each move of the PR, the customers that were allocated to facility [4] are now unassigned (see Figure 3.2.b). Hence, the first approach considered is when customers will be assigned to their nearest facility (see Figure 3.2.c). The second approach is when customers will be assigned to their most preferred facility (see Figure 3.2.d). Both approaches do not guarantee the customer's optimal allocation and semi-feasible solutions are being considered at that time. Remember that the customers that are assigned under these two approaches are the customers that are left unallocated when an facility is closed. That is, there is not a reallocation for all customers when a facility is closed and another opened. Then, when the allocation is with the second approach, this is not the optimal allocation. However, once the PR procedure has finished, the optimal resolution of the lower level is done for obtaining bilevel feasible solutions.

Regarding the EA+PRw, the two considered approaches were analyzed and it can be observed that both have good quality solutions (see Figures 3.3 and 3.4). Moreover, it can also be observed that within the first 20 generations the "most preferred" approach has a better development, but in the next generations, their performance is almost the same for the "nearest" and "most preferred" approaches. Since this behavior was observed for all instances, half of the computational experimentation was done with the "nearest" approach and the other half under the "most preferred" approach.

In order to assess the performance of the hybrid algorithm, a properly parameter identification has to be done. The parameters considered in the EA+PR are: size of the population (n), number of generations (G), and the number of tournaments (T). Initially the number of generations was of 500, but due to the convergence in EA+PR and EA (EA+PR is faster than the EA) this number was reduced, see Figures 3.5 and 3.6. The selected parameters for the EA were selected exactly as in [22]; the corresponding values are $n = 100$, $G = 150$ and $T = 5$.

In Tables 3.1, 3.2 and 3.3 the results obtained from running 10 times the EA, EA+PR and EA+PRw with each instance are shown. Table 3.1 corresponds to the results for the small-size instances, this set is composed by three subset of instances -132, 133, 134- that differ from each other on the values of distribution and fixed costs. All those instances consider four different matrices of preferences. The same procedure is used in the medium and large-size instances, their results are presented in Tables 3.2 and 3.3, respectively. Each table contains

Figure 3.2: Exemplification for reallocating the customers.

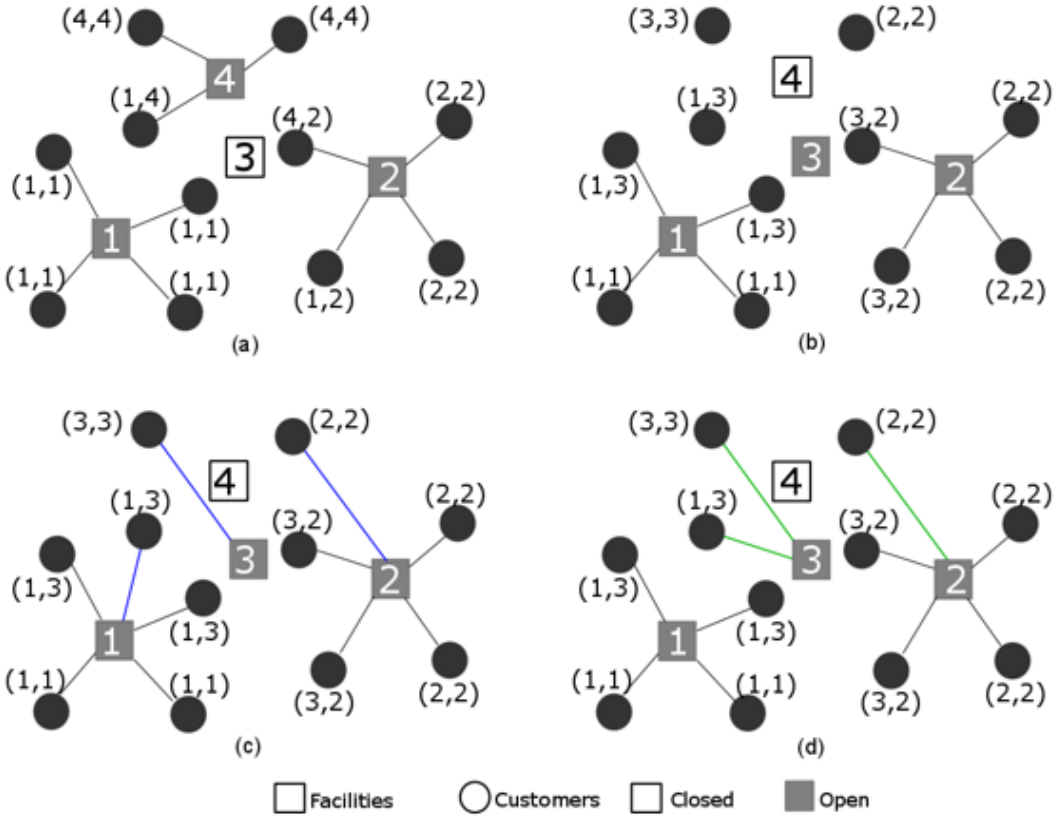
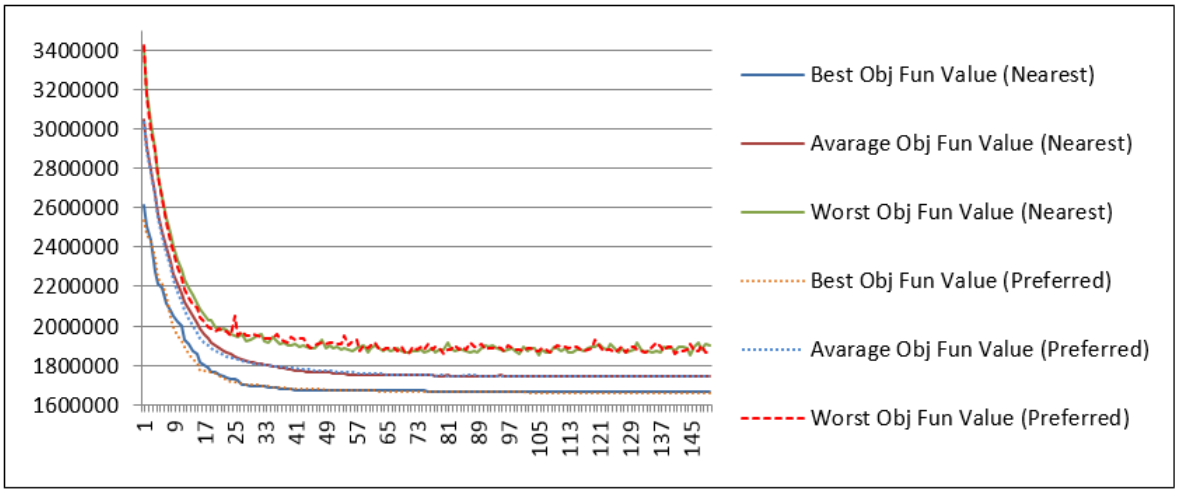


Figure 3.3: Illustrating the convergence for both approaches of EA+PRw in instance a-1 medium.



the gap between the optimal value and the best value obtained for each approach (EA, EA+PR and EA+PRw) of the 10 runs, named “best gap”. Also, the “average gap”, which is the gap between the optimal value and the average value of the 10 runs. And finally, the “average time” -in seconds- used for solving each instance is shown. The gaps described above are computed

Figure 3.4: Illustrating the convergence for both approaches of EA+PRw in instance b-2 large.

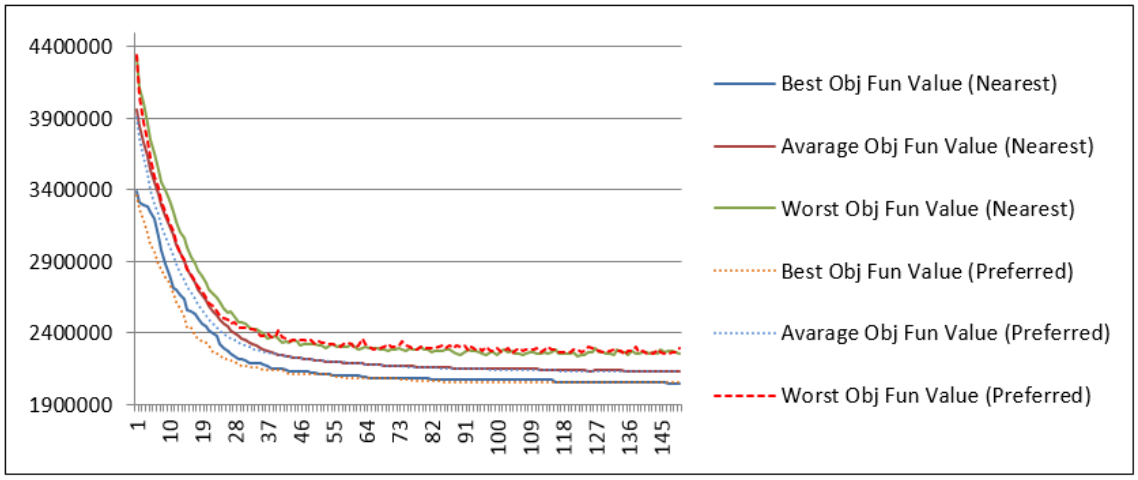


Figure 3.5: Illustrating the convergence for EA and EA+PR in instance 132-1.

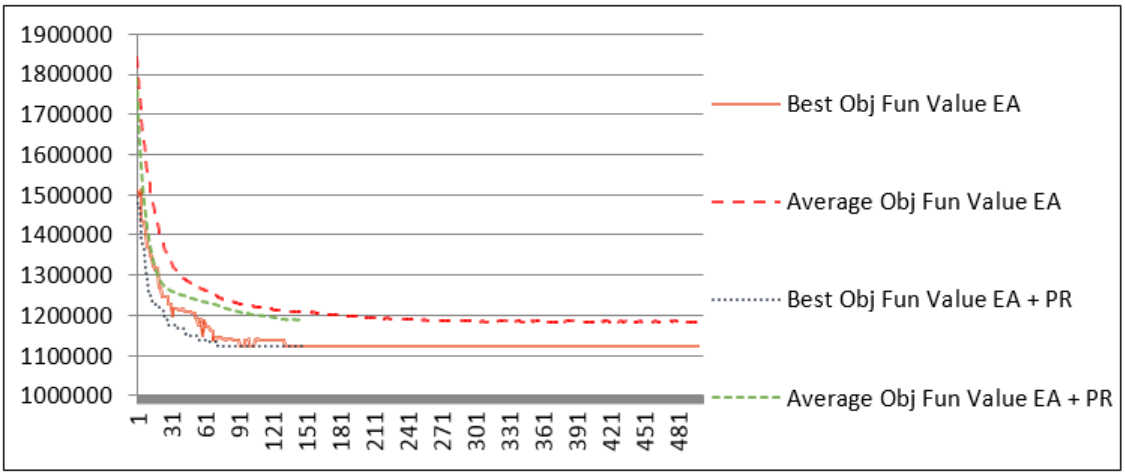
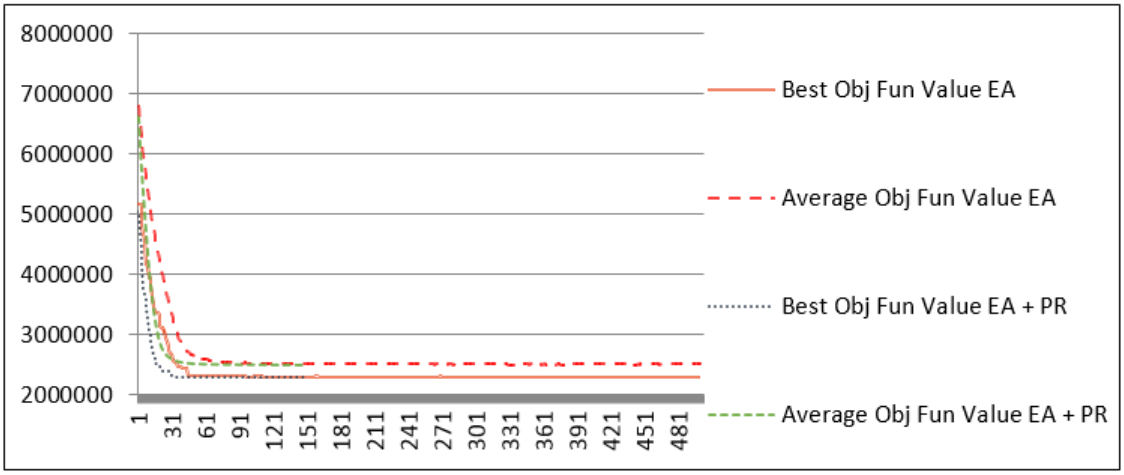


Figure 3.6: Illustrating the convergence for EA and EA+PR in instance b-1 large.



as follows: $gap = 100((x - x^*)/x^*)$, where x represents the value of objective function of a solution and x^* the value of objective function of optimal solution. It is convenient to remark that optimal values for these instances are known from previous works from the literature.

Table 3.1: Results for the small-size instances

	EA			EA+PR			EA+PRw		
Instance	**Best gap**	**Average gap**	**Time (sec)**	**Best gap**	**Average gap**	**Time (sec)**	**Best gap**	**Average gap**	**Time (sec)**
132-1	0.00%	5.41%	1.51	0.00%	5.73%	0.76	0.00%	7.91%	0.72
132-2	0.00%	4.40%	1.45	0.00%	4.14%	0.74	0.00%	4.46%	0.73
132-3	0.00%	7.81%	1.53	0.00%	6.90%	0.77	0.00%	8.59%	0.73
132-4	0.00%	4.05%	1.49	0.00%	4.12%	0.78	0.00%	4.08%	0.74
133-1	0.00%	5.13%	1.53	0.00%	4.61%	0.76	0.28%	5.03%	0.74
133-2	0.00%	5.99%	1.51	0.00%	6.99%	0.72	0.00%	6.42%	0.71
133-3	0.00%	7.45%	1.54	0.15%	6.98%	0.79	0.15%	7.96%	0.73
133-4	0.00%	2.34%	1.50	0.00%	2.04%	0.75	0.00%	2.47%	0.71
134-1	0.00%	6.72%	1.52	0.00%	6.07%	0.78	0.00%	6.78%	0.74
134-2	0.00%	6.20%	1.54	0.00%	5.67%	0.78	0.00%	6.44%	0.73
134-3	0.00%	5.72%	1.52	0.00%	5.00%	0.78	0.00%	5.69%	0.74
134-4	0.00%	4.95%	1.50	0.00%	5.07%	0.75	0.00%	5.21%	0.71

Table 3.2: Results for the medium-size instances

	EA			EA+PR			EA+PRw		
Instance	**Best gap**	**Average gap**	**Time (sec)**	**Best gap**	**Average gap**	**Time (sec)**	**Best gap**	**Average gap**	**Time (sec)**
a-1	0.09%	4.89%	2.07	0.00%	4.44%	1.04	0.09%	5.04%	0.93
a-2	0.00%	6.51%	2.08	0.00%	5.59%	1.11	0.00%	6.86%	1.00
a-3	0.04%	4.72%	2.08	0.00%	4.22%	1.10	0.00%	4.93%	0.98
a-4	0.00%	5.48%	2.07	0.00%	5.02%	1.07	0.00%	5.57%	1.00
b-1	0.00%	5.68%	2.06	0.00%	5.76%	1.08	0.00%	6.08%	1.00
b-2	0.10%	3.70%	2.07	0.00%	3.10%	1.12	0.00%	4.72%	1.08
b-3	0.00%	5.06%	2.07	0.00%	4.52%	1.10	0.49%	5.57%	1.02
b-4	0.00%	4.80%	2.04	0.02%	4.45%	1.02	0.52%	5.06%	0.98
c-1	0.26%	2.95%	2.03	0.26%	2.54%	1.10	0.26%	2.96%	1.04
c-2	0.25%	3.46%	2.04	0.00%	3.12%	1.10	0.00%	3.92%	1.04
c-3	0.00%	3.71%	2.01	0.00%	3.52%	1.06	0.00%	3.91%	1.00
c-4	0.00%	3.03%	2.04	0.00%	2.72%	1.05	0.00%	3.14%	0.99

Table 3.3: Results for the large-size instances

	EA			EA+PR			EA+PRw		
Instance	**Best gap**	**Average gap**	**Time (sec)**	**Best gap**	**Average gap**	**Time (sec)**	**Best gap**	**Average gap**	**Time (sec)**
a-1	0.00%	9.76%	3.55	0.00%	8.89%	2.07	0.00%	9.89%	1.84
a-2	0.00%	6.94%	3.49	0.00%	6.30%	2.02	0.00%	7.12%	1.77
a-3	0.00%	7.23%	3.48	0.00%	6.25%	2.09	0.00%	7.23%	1.86
a-4	0.00%	5.85%	3.61	0.00%	5.08%	2.11	0.00%	5.72%	1.85
b-1	0.77%	5.39%	3.67	0.77%	4.72%	2.35	0.87%	6.09%	2.09
b-2	1.01%	4.71%	3.65	0.29%	4.00%	2.16	0.63%	5.29%	2.05
b-3	0.00%	4.37%	3.65	0.00%	4.07%	2.13	0.86%	4.48%	2.00
b-4	0.00%	6.47%	3.71	0.00%	5.88%	2.25	1.14%	6.74%	2.08
c-1	0.00%	4.04%	3.69	0.00%	3.65%	2.22	0.00%	4.21%	2.00
c-2	0.00%	3.82%	3.65	0.00%	4.17%	2.28	0.00%	4.62%	2.05
c-3	0.00%	4.53%	3.61	0.00%	4.05%	2.12	0.00%	4.91%	2.00
c-4	0.00%	4.67%	3.69	0.00%	4.35%	2.25	0.00%	5.14%	2.05

It can be observed from Table 3.1 that there is not a difference in the best gap between EA and EA+PR. In other words, both algorithms show the same effectiveness but computational times of EA+PR are about half the computational times of EA computational time.

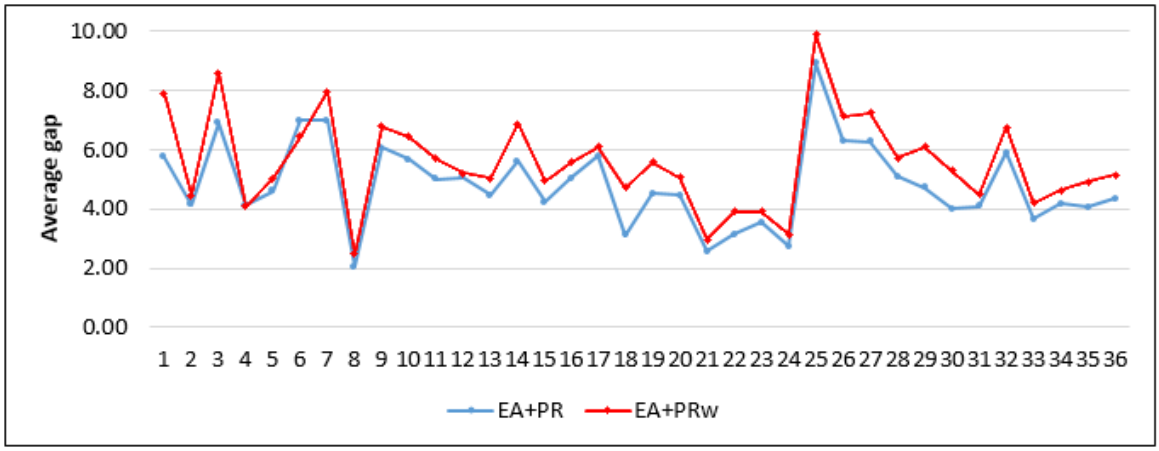
Regarding the EA+PRw, computational times are shorter than the other ones. However, in terms of quality, it is observed that EA and EA+PR provide better solutions than those provided by EA-PRw.

From Table 3.2, it can be seen again that, both EA and EA+PR seems to have the same good performance. However, the required computational time of EA+PR remains as low as half of the EA. Moreover, the average gaps in c-3 and c-4 instances are smaller values than the average gaps of EA. This means that solution quality is better for all solutions in the last population of EA-PR when it is compared with solutions in the last population of GA. On the other hand, when comparing the EA+PR with EA+PRw, it is noticed that the performance (in terms of solution quality), is worse for EA-PRw when compared with EA+PR performance. In a reasoned way, this was expected; that is, avoiding to solve the lower level problem at each move during the local search leads into a blinded exploration. Furthermore, semi-feasible solutions are being considered, which later will have to be repaired for reaching bilevel feasibility.

Finally, the behavior discussed in the above paragraphs is maintained for the large-size instances. The "best gap" and "average gap" columns do not show a significant difference between EA and EA+PR. However, as before, the computational time required for the EA+PR is smaller than computational times of EA. In addition, the EA+PR has a better performance in terms of quality than EA+PRw but the required computational times are almost the same. This is caused because in the EA+PRw when a facility is closed, the customers associated to it, need to be reallocated following a predefined criterion. Hence, a quickly reallocation is made; but the required time for solving the lower level problem using the ordered matrix of preferences is very low. Then, when the algorithm reaches its stop criterion, the cumulative time reduction is not as significant as we expected.

The average gap values obtained by both, EA+PR and EA+PRw, are plotted in Figure 3.7 for all the tested instances. The results support the ideas obtained from the discussion previously presented; that is, the performance of the EA+PRw is not as good as the EA+PR in most of the cases due to the semi-feasible solutions. It can be seen from Figure 3.7 that only twice (instances 132-4 and 133-2) the value for the EA+PRw is better than the value that corresponds for the EA+PR, but in all other instances it is worst. For example in the 132-1 and b-2 (medium-size) instances the difference between the gaps is 2.17 and 1.62, respectively.

Figure 3.7: Comparing the average gaps.



Chapter 4

p-Median Problems

4.1 Solving the p-median bilevel problem with order through a hybrid heuristic

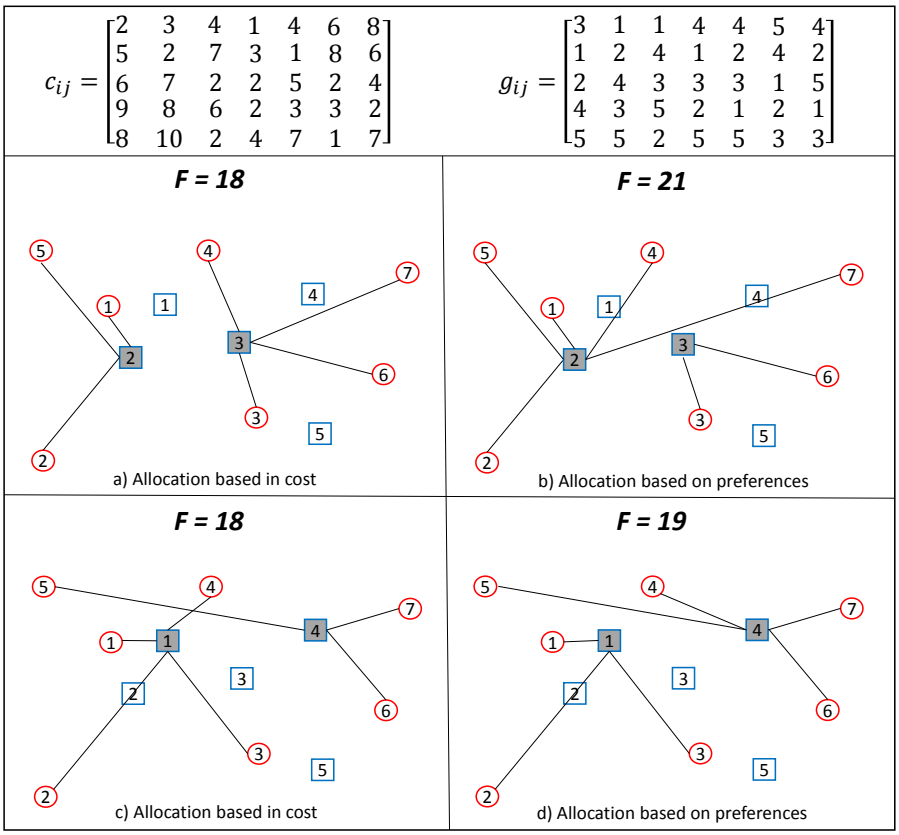
The following example illustrates the impact of introducing customer preferences in a facility location problem. Consider five potential facilities and seven customers; also, assume that two facilities must be located. The corresponding costs and customer preferences (a pre-established order set with customers preferences towards facilities) are shown in Figure 4.1. Here, i denotes facilities and j denotes customers, where $i \in I$ and $j \in J$. The c_{ij} represent the cost of supplying the entire demand of customer j from facility i. Each customer j has a preference of being served by facility i, represented by g_{ij}. Finally, p represents the number of facilities that must be located and F is the leader's objective function value.

For the example, suppose that the located facilities are $\{2, 3\}$. Using the classical the p-median problem, customers $\{1, 2, 5\}$ will be allocated to facility 2 and customers $\{3, 4, 6, 7\}$ to facility 3. The associated cost is 18 (see Figure 4.1.a). Then, using the p-median BPO problem, customers will be allocated using their preferences. In this case, customers $\{1, 2, 4, 5, 7\}$ will be allocated to facility 2 and customers $\{3, 6\}$ to facility 3; the cost of this solution 21 (see Figure 4.1.b). Now, consider an alternative solution that opens $\{1, 4\}$. In the p-median problem, customers will be partitioned in $\{1, 2, 3, 4\}$ and $\{5, 6, 7\}$ for facility 1 and 4, respectively; incurring in a cost of 18 (see Figure 4.1.c). In the problem when the preferences are considered, customers $\{1, 2, 3\}$ are allocated to facility 1 and customers $\{4, 5, 6, 7\}$ to facility 4. The cost of this decision is 19 (see Figure 4.1.d).

Despite the fact that both solutions have the same objective value in the p-median problem, the second solution is better for the problem studied in this chapter. It can be noticed that considering customer preferences negatively affects the costs, but allows to include the opinion of the customers into the decision process. When the order of the facilities coincides with the order given by the distances (or costs), the p-median BPO is reduced to the p-median problem. Hence, the p-median BPO is also NP-hard.

In location problems, it is important to consider customer opinion when opening new

Figure 4.1: Example with $I = \{1, ..., 5\}$, $J = \{1, ..., 7\}$ and $p = 2$.



facilities because in many real cases, customers are free to meet their demand from the facility they desire. This decision is usually made based on factors such as cost, preferences, predetermined contracts, and a certain coefficient of customer loyalty.

4.1.0.1 Main contributions

In this chapter, the p-median BPO, which is a variant of p-median problem that considers customer preferences, is presented. This problem is formulated as a bilevel optimization model in which the leader aims to minimize distribution costs and the follower optimizes customer preferences. This hierarchy is considered not only because of the importance of considering customer preferences but also bearing in mind that the main objective is minimizing the cost associated with distribution from the facilities to customers. In order to have a baseline for measuring the effectiveness of the proposed hybrid algorithm, two reformulations are proposed. The first reformulation is based on a classical tool for reducing bilevel problems into single-level ones. The second one is an alternative and equivalent integer model based on a set of closest assignment constraints. Moreover, a hybrid heuristic algorithm based on scatter search is developed for finding good quality solutions of the bilevel model.

Hence, one of the main contributions of this research is the study of this problem as a bilevel model for the first time. As it is mentioned in Chapter 1, Alekseeva and Kochetov [3]

and Camacho-Vallejo et al. [21] are related with this problem but the employed methodologies and research aims are quite different. Moreover, they did not studied the bilevel version of the problem but single-level models.

The other contribution is the development of a hybrid heuristic algorithm that can efficiently solve small, medium and large size instances of the bilevel version of the p-median BPO. Part of the algorithm's success is due to the proposed reduction of the neighborhood size for local search and greedy components. Also, the objective function value calculation is optimized by following the ideas described in Marić et al. [93]. A comparison between the hybrid heuristic and other algorithms shows the advantages of the proposed algorithm.

The outline of the remainder of this chapter is as follows. Section 4.1.1 describes the mathematical formulation of the problem. In Section 4.1.2, two reformulations of p-median BPO (reducing it to a single-level mixed-integer program and a binary program) are presented. In Section 4.1.3, an algorithm based on scatter search hybridized with GRASP that considers Stackelberg equilibrium is proposed. In Section 4.1.4, the results obtained from the computational experimentation by the proposed algorithm, two versions of scatter search, a genetic algorithm and the reformulations are summarized.

4.1.1 Description and mathematical formulation of p-median BPO

This problem is formulated as a bilevel optimization model. In this formulation the leader minimizes the distribution costs and the follower minimizes customer preferences. The objective of the problem to locate p facilities that minimize the distribution cost and the ordered preferences of customers in a hierarchized manner.

Let I and J be the indices of the facility location and customers, respectively, where $i \in I$ and $j \in J$. Let c_{ij} represents the cost of supplying the entire demand of customer j from facility i. Each customer j has a preference of being served by facility i, represented by g_{ij}. Finally, p represents the number of facilities that must be located. The decision variables of this bilevel model are x_{ij}, which denotes whether facility i satisfies the demand of customer j, and y_i, which denotes whether a facility is located in site i. Both decision variables are binary.

The following assumptions are also considered in the model:

- Customers a priori establish their ordered preferences for facilities as a list of numbers from 1 to $|I|$, where 1 is the most preferred facility and $|I|$ is the least preferred facility.
- Facilities have no capacity constraints associated with them, i.e., a facility can supply several customers, but each customer must be served by a unique facility by establishing the required number of facilities.

The corresponding mathematical model for the p-median BPO is as follows.

$$\min_{y} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \tag{4.1}$$

$$\text{subject to:} \sum_{i\in I} y_i = p \tag{4.2}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{4.3}$$

$$x \in \arg\min \quad \sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} \tag{4.4}$$

$$\text{subject to:} \quad \sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{4.5}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \tag{4.6}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{4.7}$$

The upper level problem is defined by (4.1)-(4.4), where (4.1) is the leader's objective function, representing the minimization of distribution costs; (4.2) indicates the total number p of facilities to be located; (4.3) establishes the binary constraints for each variable y_i; and (4.4) indicates that the variables x_{ij}, which are controlled by the follower, are determined implicitly by the optimal solution of the lower level problem. This problem is defined by (4.4)-(4.7), where the follower's objective function is (4.4), which aims to minimize customer preferences toward the facilities; (4.5) ensures that each customer is supplied by exactly one facility; (4.6) indicates that the assignment of customers is guaranteed only for the located facilities; and (4.7) indicates the binary constraints on the decision variables x_{ij}.

To ensure that p-median BPO is well defined, a unique optimal solution in the lower level problem for any fixed leader's decision y should exists. This property is maintained by the structure of preferences. The proof of this result is given in Vasil'ev et al. [136]. In the case that multiple followers' optimal responses exist, a bilevel problem needs to assume well-known optimistic or pessimistic approaches.

Notably, despite that only the follower's variables x_{ij} are involved in the leader's objective function, the leader's variables y_i are considered when solving the lower level problem. In other words, the leader's decision directly affects the distribution cost associated with the facility-customer assignment made by the follower.

4.1.2 Single-level reformulations

In this section, two reformulations of the p-median BPO are developed. The first reformulation relies on the primal-dual optimality conditions of the lower level problem. The second reformulation is an integer single-level model tantamount to p-median BPO. The fact that a subset of the problem's variables is determined implicitly by the optimal solution of another optimization problem, namely the problem given by (4.4)-(4.7), motivated us to reformulate the bilevel model as a single-level one. Hence, the resulting model could be solved by an optimizer.

4.1.2.1 Reformulation based on the primal-dual optimality conditions of bilevel problems

Note that if the leader's variables y are fixed, then the lower level involves the classical disaggregated SPLP constraints. Consequently, the single assignment property is hold (see Krarup and Pruzan [75], and Galvão [56]) assuring that a customer will be entirely supplied by its most preferred facility. Hence, binary variables x_{ij} can be relaxed without affecting the integer optimal solution. Hence, $x_{ij} \geq 0$ will be considered instead of $x_{ij} \in \{0,1\}, \forall i \in I, j \in J$. Now, the primal-dual optimality conditions of the lower level problem could be properly determined.

Let $\alpha_j, \forall j \in J$ and $\beta_{ij}, \forall i \in I, j \in J$ be the dual variables associated with the follower's primal constraints. The resulting dual problem is as follows:

$$\max_{\alpha,\beta} \quad \sum_{i \in J} \alpha_j + \sum_{i \in I} \sum_{j \in J} \beta_{ij} y_i \tag{4.8}$$

$$\text{subject to: } \alpha_j + \beta_{ij} \leq g_{ij} \quad \forall i \in I, j \in J \tag{4.9}$$

$$\beta_{ij} \leq 0 \quad \forall i \in I, j \in J \tag{4.10}$$

$$\alpha_j \text{ unrestricted} \quad \forall j \in J \tag{4.11}$$

Following the scheme presented in Marcotte et al. [90] and Camacho-Vallejo et al. [22]. The reformulation consists in replace the lower level problem by its primal and dual constraints assuring that its optimal solution is obtained. The complementary slackness constraints are chosen to guarantee lower level's optimality. Therefore, the reformulated model consists in minimizing the upper level objective function considering leader's constraints, the follower's primal and dual constraints and the corresponding complementarity slackness constraints.

Then, the non-linear reformulated model is as follows.

$$\min_{y,x,\alpha,\beta} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \tag{4.12}$$

$$\text{subject to: } \sum_{i\in I} y_i = p \tag{4.13}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{4.14}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{4.15}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \tag{4.16}$$

$$\alpha_j + \beta_{ij} \leq g_{ij} \qquad \forall i \in I, j \in J \tag{4.17}$$

$$x_{ij}(\alpha_j + \beta_{ij} - g_{ij}) = 0 \qquad \forall i \in I, j \in J \tag{4.18}$$

$$\beta_{ij}(x_{ij} - y_i) = 0 \qquad \forall i \in I, j \in J \tag{4.19}$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J \tag{4.20}$$

$$\beta_{ij} \leq 0 \qquad \forall i \in I, j \in J \tag{4.21}$$

$$\alpha_j \text{ unrestricted} \qquad \forall j \in J \tag{4.22}$$

As it is mentioned above, the problem described by (4.12)-(4.22) is obtained by considering the leader's variables y_i as parameters of the lower level problem. Constraints (4.15), (4.16), and (4.20) ensure primal feasibility; (4.17) and (4.21)-(4.22) ensure dual feasibility; and (4.18) and (4.19) force complementarity slackness. It is convenient to remark that the latter constraints are nonlinear. However, since x_{ij} and $(x_{ij} - y_i)$ are binary, (4.18) and (4.19) can be linearized using the following inequalities.

$$\alpha_j + \beta_{ij} - g_{ij} \geq -M(1 - x_{ij}) \qquad \forall i \in I, j \in J \tag{4.23}$$

$$\beta_{ij} \geq -M(1 + (x_{ij} - y_i)) \qquad \forall i \in I, j \in J \tag{4.24}$$

where M is a positive and sufficiently large constant.

Then, the primal-dual reformulation (P-D.R) is given by (4.12)-(4.17) and (4.20)-(4.24), which is a mixed-integer programming problem equivalent to p-median BPO. A similar description of this reformulation is given in Camacho-Vallejo et al. [21].

4.1.2.2 Reformulation based on the most preferred assignment constraints

In this subsection, we describe another equivalent single-level formulation for the p-median BPO. To ensure that customer preferences continue to be taken into account in the problem (the follower's problem), (4.29) is proposed. The later equation can be seen as a most preferred assignment constraint. Hence, the resulting optimization model (MPAC.R) is a binary

linear programming problem, which is expressed below:

$$\min_{y,x} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \tag{4.25}$$

$$\text{subject to: } \sum_{i\in I} y_i = p \tag{4.26}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{4.27}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J \tag{4.28}$$

$$\sum_{i\in I} x_{ij}g_{ij} \leq y_i g_{ij} + G_{max}(1 - y_i) \qquad \forall i \in I, j \in J \tag{4.29}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in I, j \in J \tag{4.30}$$

$$y_i \in \{0, 1\} \qquad \forall i \in I \tag{4.31}$$

where $G_{max} = max_{i\in I, j\in J}\{g_{ij}\} + 1$.

The key point in this single-level model relies in equation (4.29), which is a type of closest assignment constraint (see [48]). Since customers will be allocated to their most preferred facility, that decision continues to be considered in the most preferred assignment constraint (4.29). In other words, the current reformulation (4.25)-(4.31) keeps taking into account the ordered customer preferences. In detail, if facility i is located, then $G_{max}(1 - y_i)$ equals zero and (4.29) implies that among all the possible allocations, customer j will be assigned to it most preferred located facility. On the contrary, if facility i is not located, $g_{ij}y_i$ equals zero and (4.29) is relaxed since G_{max} acts as the upper bound for all ordered preferences.

Regarding the size of the models presented, note that if the problem is formulated as a bilevel model, it will have only two constraints (without considering the sign constraints), one of which is another optimization problem. In the P-D.R, the problem size increases significantly by adding $4|I||J| + |J|$ additional constraints and $|I||J| + |J|$ additional decision variables to the model. In the MPAC.R, the number of variables is maintained, but the number of constraints is increased; in the latter case, there are $2|I||J| + |J| + 1$ constraints in total. All these approaches have their own difficulties because in the bilevel model, there is an optimization problem within a constraint, which complicates its resolution. In contrast, the reformulated models have a large number of constraints that cause a nonpolynomial increase in the computational time required to solve them using an optimizer.

However, despite having a greater number of constraints, the reformulated models can be solved using software meant for optimizing mathematical programming problems. Moreover, to the best of our knowledge, no commercially available software is capable of solving a general bilevel problem. It is important to highlight that p-median BPO, P-D,R and MPAC.R are equivalent formulations for the problem herein studied.

4.1.3 Proposed algorithm's description

In this section, a hybrid heuristic algorithm based on scatter search and GRASP for solving p-median BPO is proposed. Since p-median BPO is a combinatorial problem and a subset of its variables is determined implicitly by solving another mathematical programming problem, the difficulty in solving the problem increases. The aim is that the proposed algorithm finds optimal or good quality solutions to p-median BPO within reasonable time.

4.1.3.1 Objective value calculation

In general, developing algorithms for solving bilevel problems need to consider solving the lower level problem to optimality, when the follower's variables affect the leader's objective function. As it can be seen in Equation (4.1), this is the case of the p-median BPO. Therefore, part of the algorithm's efficiency relies in the manner the lower level is solved.

For handling the semi-assignment lower level problem, the following scheme was implemented: each time an upper level solution is built, the lower level must be solved exactly. Again, owing to the high computational cost of using an optimizer for solving the lower level problem, the methodology proposed in [93] is considered.

From a game theoretic point of view, the proposed algorithm considers a classical non-cooperative Stackelberg game, which reflects the competitive case of a leader seeking to minimize distribution costs and a follower seeking to minimize customer preferences. During the iterative process for constructing solutions for the bilevel problem, consideration of the optimal follower's response to a leader's decision reflects the aim of finding Stackelberg's equilibrium. This equilibrium is achieved when the leader selects its best decision considering the optimal follower reaction. This is the most appropriate approach used for solving bilevel problems.

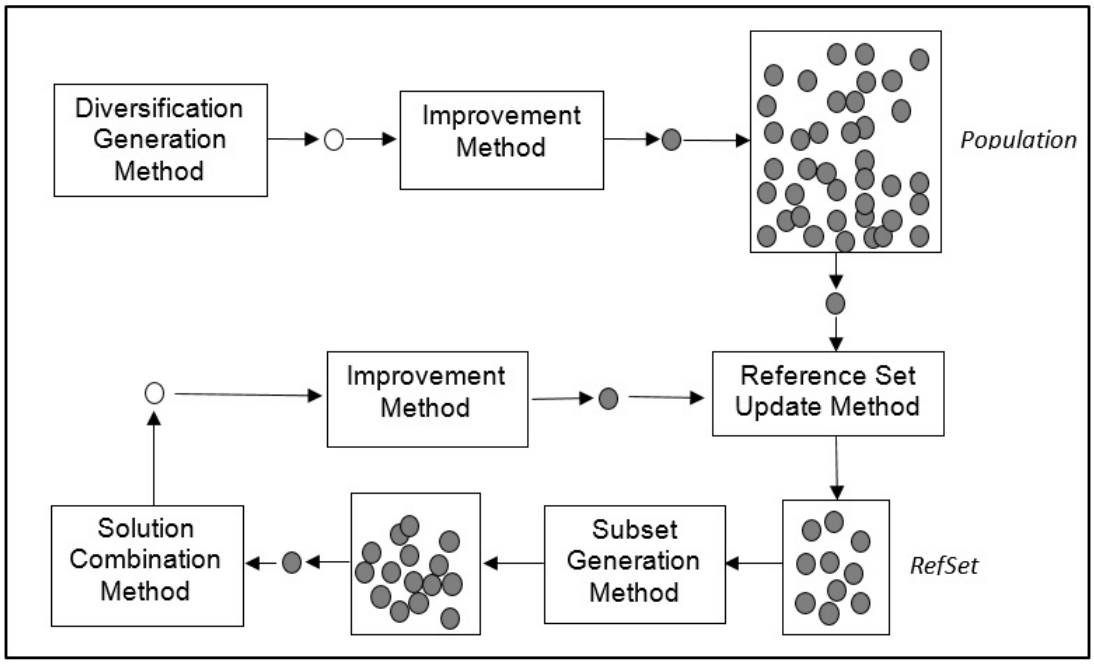
4.1.3.2 Scatter Search

The scatter search metaheuristic is applied to the p-median BPO owing to its versatility and good performance in solving complex problems. Scatter search is a population-based algorithm that has some similarities to genetic algorithms. The main differences are in terms of the use of systematic strategies rather than random operators and in the number of solutions that conform to the population (reference set). For a detailed description of this metaheuristic and its applications, see [81].

This metaheuristic starts by creating a set of solutions called the $Population$ consisting of P_{size} elements. From which b elements are extracted from this $Population$ to form the reference set, which should contain $b/2$ good quality solutions and $b/2$ diverse solutions. Then, a set that contains all possible combinations of pairs of solutions belonging to the reference set is created. Once the new solution set is created, it is input to the improvement method, which is divided into two phases: 1) the feasibility phase, which seeks to convert an infeasible solution into a feasible one because the combination method does not guarantee that the solutions developed are feasible, and 2) the improvement phase, which attempts to improve

the current solution by a local search procedure. This set of combined solutions are used to update the reference set. The algorithm's stop criterion is met when there are no new solutions to be generated, that is, when the reference set is not updated. Figure 4.2 shows a schematic of scatter search.

Figure 4.2: Schematic of the scatter search algorithm.



An adapted and detailed description of the five methods shown in Figure 4.2 for solving p-median BPO is presented. Note that because scatter search is a metaheuristic, the different components of the algorithm may vary and need to be specially adapted according to the problem at hand.

Diversification Generation Method

Before the description, it is convenient to highlight that a solution in the algorithm corresponds to a leader's decision y. A feasible solution is constructed iteratively by locating a facility in each iteration until the solution reaches the desired cardinality, i.e., until p facilities are located. Then, in order to generate a pool of P_{size} solutions called $Population$, two different approaches are implemented. The first approach consists in a random construction; that is, p random facilities are included into the current solution. The second approach is a greedy construction, in which in order to decide the facilities that are going to be located in a particular solution, a greedy function that measures the local contribution of each element to the partial solution is considered. The greedy function is defined as the sum of the distribution costs associated with locating a specific facility. In order to create different greedy solutions, the first facility included in the solution will be selected in a random fashion. By doing this, the local contribution associated with the remaining facilities will be modified.

Reference Set Update Method

After creating the initial $Population$, a subset named reference set ($RefSet$) with b solutions is created. $RefSet$ must contain good quality and diverse solutions. To meet these criteria, the best $b/2$ solutions in $Population$ are included in $RefSet$. The solution quality is based on the leader's objective function value. The other $b/2$ solutions are also selected from $Population$, but maximizing the minimum distance of the solutions already included in the $RefSet$ to maintain diversity. Because of the representation of the leader's solutions, a metric of distance is considered as the sum of the absolute values of the differences among its variables. That is, $d(y, y') = \sum_{i \in I} |(y_i - y'_i)|$, where y and y' are solutions of the sets $Population$ and $RefSet$, respectively. Then, $RefSet$ is updated according to the following criterion: each combined solution can enter this set if and only if it improves the worst objective value associated with the solutions already in the set. The method stops when no new solutions can be included.

Subset Generation Method

The intention of this method is to generate subsets of fixed size that contain solutions belonging to $RefSet$ applying the solution combination method. In our case, we consider all subsets of size 2. In other words, all possible pairs of solutions are explored. Since scatter search is an iterative algorithm, we keep a record about the combinations performed in order not to combine repeated pairs. The algorithm stops when all the possible pairs generated in the subset generation method has been combined. Note that if the $RefSet$ do not vary from one iteration to another, no new unexplored pairs will be available.

Solution Combination Method

This method combine pairs of solutions for obtaining new solutions. The method considers a score associated with each component of a pair of combined solutions. Without loss of generality, consider a pair of solutions formed by solutions y^1 and y^2. Let y_i be the i-th component of a given solution y. For each i in the new combined solution, the score is computed as follows:

$$score_i = \frac{LOF^1 y_i^1 + LOF^2 y_i^2}{LOF^1 + LOF^2} \tag{4.32}$$

where LOF^1 and LOF^2 indicate the leader's objective function value associated with solutions y^1 and y^2, respectively. Then, a vector of appropriate dimensions is filled with $rand_i$ random numbers, where $0 \leq rand_i \leq 1$. If $rand_i \leq score_i$, the corresponding component of the new combined solution will take the value of 1; otherwise, it will take the value of 0.

In order to illustrate the combination method, consider the example shown in Figure 4.3. The purpose of the formula given by (4.32), is to favor those facilities that are located in both solutions (see the 8th component of y^1 and y^2). In this case, the corresponding score is 1 ($score_8 = 1$), and that facility will remain located in the combined solution. Analogously, if a facility is not located in both solutions, then it cannot be located in the combined solution.

Figure 4.3: Combination method.

$y^1 = [0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0]$ $LOF^1 = 1320239.75$

$y^2 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0]$ $LOF^2 = 1329914.25$

$score = [0.502\ 0.000\ 0.000\ 0.499\ 0.000\ 0.499\ 0.000\ 1.000\ 0.502\ 0.000]$

$rand = [0.308\ 0.152\ 0.520\ 0.179\ 0.780\ 0.252\ 0.260\ 0.180\ 0.329\ 0.845]$

$y_{combined} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0]$

Improvement Method

This method consists of two phases: feasibility and improvement. The feasibility phase ensures that the combined solution has exactly p located facilities. A solution may be infeasible after the combination method because it may have more or fewer located facilities than p. For example, this fact can be appreciated in the combined solution shown in Figure 4.3 in which $p = 3$ and $y_{combined}$ contains five located facilities. To obtain a feasible solution, this phase may proceed in two ways: locate or close the necessary facilities until the number p is reached. The criterion followed for locating facilities is the same as that in the initial construction phase. In contrast, for removing facilities, the impact of the leader's objective function is computed after removing a specific facility. The most convenient facility, in terms of the leader's objective value, is removed. Once the combined solution is feasible, it enters the improvement phase. Here a local search is applied, which involves interchanging all facilities in the current solution with the candidate facilities (e.g. [132]), that is, a closed facility is exchanged for an open. The interchange that leads to the best reduction in the distribution cost is performed -that is, the best improvement strategy. Then, the local search continues exploring the subsequent neighborhoods until no further improvement is achieved.

4.1.3.3 GRASP

The greedy randomized adaptive search procedure (GRASP) is a metaheuristic designed to solve combinatorial optimization problems (e.g., [114]). Each iteration of the heuristic consists of two phases: 1) a constructive phase, where a good feasible solution is found, and 2) a local search phase which seeks to improve the current solution.

In the first phase, a solution is build iteratively adding an element at each iteration. In order to select these elements, a greedy function that measures the local contribution of each element to the partial solution is considered. Based on the contribution of all candidate elements, a restricted list of candidates is created. The procedure is adaptive because the costs associated with the candidate elements are updated in each iteration in order to reflect the

changes caused by the selection of the previous elements. Moreover, the process is randomized because of the random selection of items from the restricted list of candidates. A local search method is always beneficial to improve the initial solution. In each iteration in the local search, a portion of the solution is replaced with the most convenient part within a neighborhood. The procedure ends when no there is an improvement in the neighborhood.

Constructing a solution

The details of the construction phase are given below and shown in Algorithm 4.1.1. To decide the facilities that are going to be located in a particular solution, the same greedy function described in Section 4.1.3.2 is considered.

A crucial part of the construction phase is the selection of a candidate facility as a part of the current solution. First, it is important to remember that for evaluating each candidate's inclusion in the solution, the lower level needs to be solved optimally. Since evaluation of all possible candidates is computationally expensive, only a percentage of the total number of potential candidate facilities ($N < |I|$) is considered. That is, a reduction on the neighborhood size is proposed. Thus, candidates in this reduced list are chosen randomly. Then, each candidate is added into the solution (y) in an independent manner, the lower level is solved for obtaining the customer allocation and the leader's cost is computed in $Compute\_change$. Now, the minimum and maximum leader's costs (c_{min} and c^{max}, respectively) for the current reduced candidate list can be determined. Hence, the threshold value $c_{min} + \alpha(c^{max} - c_{min})$ is calculated, where $0 \leq \alpha \leq 1$ indicates the desired degree of randomness. For each candidate facility in the reduced list, we verified whether the associated leader's cost is lower than the threshold value; if so, that candidate facility is introduced in the restricted candidate list (RCL). Then, the selection of elements in the RCL is randomly performed. The selected facilities are included in the current solution and the entire process is repeated until a stopping criterion is met; in this case, until the solution has obtained p located facilities.

Algorithm 4.1.1 Construction procedure in GRASP.

procedure CONSTRUCTION( )
  $RCL \leftarrow \emptyset$
  $y \leftarrow \emptyset$
  while $|y| < p$ **do**
    $Candidates \leftarrow I$
    $count = 0$
    repeat
      Randomly select $e \in Candidates$
      $\epsilon(i) \leftarrow e$
      $y' \leftarrow y \cup \{e\}$
      $x \leftarrow Solve\_lower\_level(y')$
      $\Omega(i) \leftarrow Compute\_change(y')$
      $y \leftarrow y'$
      $Candidates \leftarrow Candidates \setminus \{e\}$
      $count = count + 1$
    until $count = N$
    $c^{max} \leftarrow max\{\Omega(k) : k = 1, ..., N\}$
    $c_{min} \leftarrow min\{\Omega(k) : k = 1, ..., N\}$
    $RCL \leftarrow \{k = 1, ..., N : \Omega(k) \leq c_{min} + \alpha(c^{max} - c_{min})\}$
    Randomly select $k \in RCL$
    $y \leftarrow y \cup \{\epsilon(k)\}$
    $I \leftarrow I \setminus \{\epsilon(k)\}$
  end while
  return y
end procedure

Intensification of the initial solution

Because it is not possible to ensure that a good solution for the problem will be generated by GRASP in the construction phase, it is convenient to conduct a local search. Due to the cardinality of the solution is fixed (p facilities), the local search only involves interchanges between a located facility with an unlocated one, i.e., interchange of a facility that is in the current solution with one that is not. This local search procedure guarantees feasibility of the solution and stops when no further improvement is possible. The main difference with the improvement method used in scatter search is that, in GRASP a neighborhood reduction is considered for the local search. Algorithm 4.1.2 shows the pseudocode of the proposed local search.

As it is mentioned above, the evaluation of the leader's objective function for each possible solution explored in the local search, the customers-facilities allocation problem needs to be solved. For reducing this effort, we proceed in a similar manner to that in the GRASP construction phase, meaning we examine only a fraction of the potential candidate facilities. This fraction is denoted by N. One half of the selected potential candidates is chosen based

on their quality (C_1) and the other half is selected randomly (C_2). The quality of the i-th candidate facility is measured using the maximum suitability ratio, that is, $r^{max}(i) = max_j\{r_{ij}\}$, where $r_{ij} = (c^{max}(j) - c_{ij})/g_{ij}$ and $c^{max}(j) = max_i\{c_{ij}\}$ represents the maximum distribution cost for each customer j. This ratio reflects the suitability of introducing a facility into the solution, i.e., for a small $\delta > 0$, $c^{max}(j) - c_{ij} \leq \delta$ implies that for customer j, facility i corresponds to a high distribution cost regardless of the value of g_{ij}, which represents the customer preferences. In the case that $c^{max}(j) - c_{ij} >> 0$, preference value g_{ij} plays a key role in the selection. Clearly, lower values of g_{ij} are desired (most preferred). Then, for each potential candidate facility i, the maximum value $r^{max}(i)$ is computed and its corresponding quality is measured. After selecting the potential candidates, a movement of the local search (interchange) is performed. Note that the potential candidates must be selected again for each improved solution until the local search stops.

Algorithm 4.1.2 Local search procedure.

procedure LOCAL SEARCH
  For a given solution y
  $flag = true$
  while $flag = true$ **do**
    $I' \leftarrow I \setminus \{facilities \quad in \quad solution \quad y\}$
    $C_1 \leftarrow \emptyset$
    $C_2 \leftarrow \emptyset$
    $B \leftarrow 0$
    $flag = false$
    for $j = 1, ..., |J|$ **do**
      $c^{max}(j) = argmax\{c_{ij} : i \in I'\}$
    end for
    for all $i = 1, ..., |I'|, j = 1, ..., |J|$ **do**
      $r_{ij} = (c^{max}(j) - c_{ij})/g_{ij}$
    end for
    for $i = 1, ..., |I'|$ **do**
      $r^{max}(i) = argmax\{r_{ij} : j \in J\}$
    end for
    for $k = 1, ..., N/2$ **do**
      $e_k^1 = argmax\{i \in I' \setminus (C_1 \cup C_2) : r^{max}(i)\}$
      $C_1 \leftarrow C_1 \cup \{e_k^1\}$
      Randomly select $e_k^2 \in I' \setminus (C_1 \cup C_2)$
      $C_2 \leftarrow C_2 \cup \{e_k^2\}$
    end for
    while Exists an unexplored pair $(y(i) \in y, e_k \in C_1 \cup C_2)$ **do**
      $y' \leftarrow Interchange(y(i), e_k)$
      $x \leftarrow Solve\_lower\_level(y')$
      $c(y') \leftarrow Compute\_change(y')$
      $B \leftarrow max\{0, c(y) - c(y')\}$
      if $B > 0$ **then**
        $y \leftarrow y'$
        $flag = true$
      end if
    end while
  end while
  return y
end procedure

4.1.3.4 Hybrid algorithm

Hybrid algorithms have attracted the attention of the researches over the last years (see [130]). The aim of hybridizing algorithms is to enhance the advantages given by particular components of each algorithm. In this algorithm, we are combining two metaheuristics: scatter

search and GRASP (SS-GRASP). Some cases of success when combine these two metaheuristics can be found in [58], [123], [74], [33], [95], [23]. In the proposed hybrid algorithm, GRASP is used in the diversification generation method, that is, for constructing good quality and diverse solutions. The GRASP considered in scatter search is depicted in Algorithm 4.1.3.

Algorithm 4.1.3 Pseudocode of GRASP for the diversification generation method.

```
1: procedure GRASP
2:     Population ← ∅
3:     k ← 1
4:     y_k ← ∅
5:     while k ≤ P_size do
6:         y_k ← Construction(y_k)
7:         y_k ← LocalSearch(y_k)
8:         Population ← Population ∪ {y_k}
9:         k ← k + 1
10:    end while
11:    return Population
12: end procedure
```

It is important to notice that the GRASP used for the diversification generation method considers the reduced neighborhood scheme proposed above. The reference set update method, the subset generation method and the solution combination method are the same explained in Section 4.1.3.2. Then, for the improvement method the same local search implemented for GRASP is considered. A pseudocode of the proposed heuristic is depicted in Algorithm 4.1.4.

We can see that the initial population ($InitialSet$) of the scatter search is generated through a GRASP procedure, then a local search ($LocalSearch$) is applied to this population. Then, a subset of the population, called a reference set or $RefSet$, is created ($ReferenceSet\ UptadeMethod$). Then, all possible subsets of size two of the $RefSet$ are formed ($Subset\ GenerationMethod$). Subsets are passed to a combination method ($SolutionCombination\ Method$) and then to a local search ($LocalSearch$). Finally, the $RefSet$ is updated ($Reference\ SetUptadeMethod$) and the algorithm ends when there are no pairs to be explored.

Algorithm 4.1.4 Pseudocode of the hybrid algorithm (SS-GRASP).

procedure HYBRID ALGORITHM
  $InitialSet \leftarrow GRASP(P_{size})$
  $RefinedSet \leftarrow \emptyset$
  for $(y_k \in InitialSet)$ **do**
    $RefinedSet \leftarrow LocalSearch(y_k)$
  end for
  $RefSet \leftarrow ReferenceSetUptadeMethod(RefinedSet)$
  while (unexplored pairs exist) **do**
    $Subsets \leftarrow SubsetGenerationMethod(RefSet)$
    $SolutionSet \leftarrow \emptyset$
    for $(subset_k \in Subsets)$ **do**
      $\bar{y}_k \leftarrow SolutionCombinationMethod(subset_k)$
      $SolutionSet \leftarrow LocalSearch(\bar{y}_k)$
    end for
    $RefSet \leftarrow ReferenceSetUptadeMethod(RefSet, SolutionSet)$
  end while
  return $RefSet$
end procedure

4.1.4 Computational experiments

Computational testing was conducted to analyze both the capability of optimally solving the reformulations and the performance of the developed heuristics algorithms. For the computational experiments, a set of 36 instances reported in [26] and a set of 36 larger instances used in [22] were adapted by omitting the fixed costs and fixing a p value[a]. Within the first set of instances, three different subsets of 12 instances were tested, which can be classified as small, and medium-size instances. The small-size instances consisted of 50 facilities and 50 customers (50×50), and the medium-size instances were 50×75 and 75×100. For this set of instances, the value of p was selected as the same reported in [26]. Regarding the second set of instances, the structure is the same as the one described above. This is, three subsets are considered, in which each subset contains 12 instances of size 100×1000, 300×1000 and 500×1000, respectively. The number of facilities to be located in the larger-size instances was selected in a random way between 5% and 10% of the potential facilities ($|I|$).

Experiments were performed on a PC with a 3.1 GHz Intel(R) Core i5-4440 processor and 8 GB RAM. The codes of the heuristic algorithms were implemented in Visual Studio 2013 using C++. CPLEX 12.6.1 was also used for the experiments pertaining the reformulations.

The computational testing can be divided in two main parts. First, the instances were solved with CPLEX using the three single-level formulations of p-median BPO: (i) P-D.R (ii) MPAC.R, and (iii) an adaptation of the single-level integer formulation proposed in [135],

[a]The instances are available at: http://www.fcfm.uanl.mx/es/PCOM/instancias

referred as V&K.R. In order to adapt the latter formulation, fixed costs were omitted from the leader's objective function and the constraint that forces to locate exactly p facilities is included; the remaining structure of the problem is the same. The second part of the computational experimentation was concerned with the proposed heuristic algorithm and the other heuristics developed for comparison; that is, scatter search with random construction (SSr), scatter search with greedy construction (SSg), and a Genetic algorithm (GA). Parameter tuning was conducted for improving their performance. To measure the efficiency of the tested algorithms, they were applied to solve the same set of instances as those considered in the reformulations.

It is natural to compare a hybrid heuristic against one heuristic that is being combined to conform it. The reason for this comparison is to measure the impact that the hybridization has on the incumbent solution. Moreover, based on [98], which is a survey regarding metaheuristic approaches for solving the p-Median problem, a Genetic Algorithm (GA) similar than the one proposed in [42] was implemented. Their same solution coding and standard genetic operators as the ones described that paper were used in our implementation. The main motivation for selecting a GA is that it is a population-based metaheuristic as the hybrid proposed heuristic. In the same manner as that for the algorithms described in Section 4.1.3, the lower level was re-solved after a facility was included in the solution within the GA.

4.1.4.1 Numerical results

A parameter tuning for the tested heuristic algorithms is conducted. Since the proposed heuristic hybridizes scatter search and GRASP, we decided to calibrate its parameters first. This effort will serve us to calibrate SSr, and SSg. Based on the results, $\alpha = 0.4$ and $N = 6$ (number of candidates) were selected. The parameters considered in SSr and SSg are set as the same of the ones used in the SS-GRASP. However, it is important to highlight that the full neighborhood is explored in their improvement phase. Regarding the GA, its parameters were tuned replicating the same methodology describe in [22]. The selected parameters are as follows: the population size was fixed to 100 and the probabilities to enter into the crossover and mutation were set to 0.75 and 0.15, respectively. Instead of considering a maximum number of generations as a stop criterion, the GA was run until the same time needed for the hybrid heuristic was reached. This stop criterion will lead us to directly compare the performance of both algorithms.

The considered instances were solved and the results are summarized in Tables 4.2-4.1.3. The structure of tables corresponding to each different set of instances is as follows: the first column lists the label that indicates the corresponding instance, the second column lists the number p of located facilities, and the third column lists the optimal values of the leader's objective function. Then, for each single-level reformulation its optimality gap and the required time (in seconds) for each instance is shown. This gap is the one displayed by CPLEX when the 6 h set as maximum time was reached. Furthermore, the second part of the tables is devoted to the tuned heuristics. The results can be divided in five parts, namely, the results of the SSr, SSg, SS-GRASP, and GA. Columns %B, %A and %W indicate the percent deviation of the best value reached for the algorithm, average value between the runs

and worst value between the runs with respect to the optimum value or better known value, respectively. Also, the column t(s) shows the average time required (in seconds) for solving each instance. Note that the GA was stopped when it reaches the same time as the hybrid algorithm. In order to give an idea about the behavior of the solution methods in the same size instances, the average value for the corresponding column is included in the last row. As noticed before, there exists a certain degree of stochasticity in the four approaches; hence, each instance was run 10 times. For the first set of instances, the reference value used for computing the latter percent deviations coincides with the optimum given by the single-level reformulations.

Table 4.1: Comparison for the 50 $\times$ 50 instances.

			P-D.R		MPAC.R		V&K.R		SSr				SSg				SS-GRASP				GA		
Inst	**p**	**Opt**	**%**	**t (s)**	**%**	**t (s)**	**%**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**
132_1	8	1022749	0.00	253.99	0.00	1.93	0.00	1.78	0.00	0.29	1.45	1.63	0.00	1.18	11.85	2.30	0.00	0.14	1.45	0.44	3.94	6.48	11.61
132_2	9	1056019	0.00	700.87	0.00	2.08	0.00	1.86	2.65	2.66	2.78	1.29	0.00	2.42	5.55	1.98	0.00	0.51	2.02	0.49	3.35	5.47	7.60
132_3	6	1083801	0.00	1294.67	0.00	2.22	0.00	2.38	0.00	0.00	0.00	1.16	0.00	0.00	0.00	1.43	0.00	0.40	3.98	0.34	0.00	3.79	6.17
132_4	5	986779	0.00	34.57	0.00	1.82	0.00	1.60	0.00	0.00	0.00	0.90	0.00	0.00	0.00	1.33	0.00	0.14	1.39	0.32	0.00	0.28	1.39
133_1	7	998271	0.00	94.85	0.00	1.16	0.00	1.27	0.00	0.13	1.32	1.29	0.00	0.39	3.88	1.74	0.00	0.13	1.32	0.39	0.00	0.00	0.00
133_2	5	965442	0.00	7.60	0.00	1.17	0.00	1.06	0.00	0.00	0.00	1.05	0.00	0.00	0.00	1.21	0.00	0.36	3.61	0.35	0.00	5.52	10.92
133_3	6	1083831	0.00	113.73	0.00	1.92	0.00	1.71	0.00	0.59	3.49	1.14	0.00	1.67	2.39	1.36	0.00	0.23	2.30	0.41	2.39	4.10	6.81
133_4	9	940194	0.00	966.76	0.00	1.69	0.00	1.49	0.00	0.00	0.00	1.08	0.00	0.29	1.98	1.72	0.00	0.02	0.20	0.41	0.00	0.70	1.64
134_1	4	1104639	0.00	86.15	0.00	3.07	0.00	3.21	0.00	0.00	0.00	0.86	0.00	0.00	0.00	1.03	0.00	0.43	4.30	0.46	0.00	0.43	4.30
134_2	7	971632	0.00	140.29	0.00	1.22	0.00	1.32	0.00	1.78	2.96	1.13	0.00	0.89	2.96	1.63	0.00	0.21	2.15	0.35	0.00	1.21	3.25
134_3	6	1043409	0.00	358.64	0.00	3.47	0.00	3.56	0.00	1.41	1.75	1.34	0.00	0.33	1.63	1.69	0.00	0.12	0.29	0.37	0.00	0.63	2.32
134_4	3	1160465	0.00	16.50	0.00	2.28	0.00	2.19	0.00	0.00	0.00	0.78	0.00	0.00	0.00	0.95	0.00	0.12	1.18	0.37	0.00	0.00	0.00
		Average	0.00	339.05	0.00	2.00	0.00	1.95	0.22	0.57	1.15	1.14	0.00	0.60	2.52	1.53	0.00	0.23	2.02	0.39	0.81	2.38	4.67

Table 4.2: Comparison for the 50 $\times$ 75 instances.

			P-D.R		MPAC.R		V&K.R		SSr				SSg				SS-GRASP				GA		
Inst	**p**	**Opt**	**%**	**t (s)**	**%**	**t (s)**	**%**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**
a_1	7	1176842	32.64	21600.00	0.00	17.03	0.00	16.62	0.00	2.14	3.60	1.67	0.00	1.18	2.96	2.65	0.00	0.59	2.96	0.68	2.96	4.84	6.45
a_2	6	1231027	29.27	21600.00	0.00	19.25	0.00	21.23	0.00	0.02	0.08	1.57	0.00	0.05	0.08	2.73	0.00	0.01	0.08	0.70	0.08	2.53	5.57
a_3	7	1138492	0.00	6894.26	0.00	19.98	0.00	12.64	0.00	0.34	2.20	1.89	0.00	0.24	1.22	3.91	0.00	0.10	0.52	0.69	0.00	1.29	2.61
a_4	5	1274992	0.00	4752.11	0.00	13.85	0.00	15.51	0.00	0.11	1.09	1.67	0.00	0.11	1.09	1.09	0.00	0.22	1.09	0.64	0.00	0.44	1.09
b_1	8	1036201	22.55	21600.00	0.00	16.29	0.00	16.01	0.00	3.20	4.48	1.98	0.00	2.73	4.71	2.99	0.00	0.19	0.62	0.75	4.48	4.88	5.33
b_2	9	1092675	35.11	21600.00	0.00	25.31	0.00	27.54	0.00	0.58	2.24	2.28	0.00	0.14	1.38	3.77	0.00	0.52	1.04	0.82	1.38	2.93	4.79
b_3	9	976995	0.00	4233.79	0.00	11.13	0.00	11.33	0.00	2.18	7.12	2.30	0.00	0.54	5.36	3.49	0.00	0.63	2.20	0.81	0.00	3.29	7.12
b_4	9	990407	26.11	3589.00	0.00	9.65	0.00	9.68	0.00	0.00	0.00	1.76	0.00	0.00	0.00	2.95	0.00	0.26	0.85	0.67	1.78	3.88	5.62
c_1	11	1097176	41.02	21600.00	0.00	19.63	0.00	19.26	0.00	1.06	1.92	2.28	0.00	0.81	1.92	3.77	0.00	0.31	0.63	0.83	0.63	3.20	7.14
c_2	10	1038938	0.00	9401.08	0.00	10.64	0.00	10.66	0.00	0.52	2.17	2.72	0.00	0.97	5.31	3.61	0.00	0.30	1.25	0.72	2.28	5.10	8.83
c_3	12	977987	0.00	8664.12	0.00	7.69	0.00	7.31	0.00	0.00	0.00	2.15	0.00	0.00	0.00	5.21	0.00	0.27	0.98	0.82	0.00	6.15	12.67
c_4	11	1129174	28.93	21600.00	0.00	14.30	0.00	23.23	0.00	1.32	1.58	2.15	0.00	2.08	4.01	3.55	0.00	0.47	1.58	0.79	0.00	4.01	6.22
		Average	17.97	13927.86	0.00	15.40	0.00	15.92	0.00	0.96	2.21	2.04	0.00	0.74	3.34	3.31	0.00	0.32	1.15	0.74	1.13	3.55	6.12

Table 4.3: Comparison for the 75 $\times$ 100 instances.

			P-D.R		MPAC.R		V&K.R		SSr				SSg				SS-GRASP				GA		
Inst	**p**	**Opt**	**%**	**t (s)**	**%**	**t (s)**	**%**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**
a_1	4	1783375	35.46	21600.00	0.00	102.03	0.00	91.46	0.00	0.67	6.67	6.22	0.00	0.52	1.58	6.20	0.00	1.57	3.98	1.58	0.00	3.42	9.69
a_2	3	2078834	0.00	3498.09	0.00	77.59	0.00	80.60	0.00	0.00	0.00	4.66	0.00	0.00	0.00	5.10	0.00	0.17	0.58	1.43	0.00	0.16	0.97
a_3	3	2041922	0.00	8543.12	0.00	79.98	0.00	82.60	0.00	0.03	0.33	5.52	0.00	0.07	0.33	5.28	0.00	0.13	0.33	1.60	0.00	0.62	1.29
a_4	4	1914302	31.05	21600.00	0.00	173.27	0.00	145.40	0.00	0.22	0.48	5.41	0.00	0.16	1.67	7.42	0.00	0.32	1.60	1.67	0.00	1.65	2.60
b_1	8	1535299	37.88	21600.00	0.00	893.49	0.00	833.52	0.00	1.76	5.15	9.48	0.00	1.76	3.47	9.58	0.00	0.36	0.89	3.54	1.75	4.72	6.57
b_2	8	1605970	47.52	21600.00	0.00	944.19	0.00	1032.20	0.00	0.59	2.49	9.07	0.00	0.25	1.48	9.22	0.00	0.21	0.70	2.00	1.13	3.46	6.25
b_3	8	1602332	39.14	21600.00	0.00	613.04	0.00	680.95	0.00	1.16	3.22	7.58	0.00	1.91	4.52	8.58	0.00	0.39	1.30	2.46	3.52	6.42	9.19
b_4	9	1366849	0.00	18412.37	0.00	555.87	0.00	503.48	0.00	1.92	4.81	9.33	0.00	3.10	5.37	9.44	0.00	0.63	1.57	2.04	4.81	9.06	10.70
c_1	6	1642970	0.00	9686.41	0.00	166.55	0.00	157.08	0.00	0.35	1.74	7.16	0.00	0.17	1.74	7.39	0.00	0.17	0.56	1.78	0.93	4.47	6.56
c_2	11	1402881	48.45	21600.00	0.00	632.06	0.00	636.93	0.55	1.40	2.15	9.01	0.55	1.82	2.97	13.90	0.00	0.64	1.06	2.25	5.24	7.82	9.90
c_3	8	1538034	43.20	21600.00	0.00	490.82	0.00	470.83	0.00	0.17	1.71	7.92	0.00	0.74	2.45	9.57	0.00	0.17	0.84	1.98	4.35	5.77	7.26
c_4	4	1465545	42.40	21600.00	0.00	481.26	0.00	465.14	0.00	0.66	6.59	9.58	0.00	0.23	1.13	10.80	0.00	0.40	1.13	2.12	0.74	3.85	8.83
		Average	27.09	17744.98	0.00	434.18	0.00	431.68	0.05	0.74	2.95	7.58	0.05	0.89	2.23	8.54	0.00	0.43	1.21	2.04	1.87	4.29	6.65

For the first set of instances, let us discuss the results obtained from the single-level reformulations. From Tables 4.1, 4.2, and 4.3, we can see that P-D.R was unable to find the optimal value in 14 of the 36 instances within the 6 h time limit. In contrast, both MPAC.R and V&K.R were able to optimally solve the 36 small and medium-size instances. However, there is no evidence to guarantee that MPAC.R requires less time than V&K.R when solving

the integer reformulations. The time required to obtain an optimal solution for a 50×50 instance varies between 1 and 3.5 s, that for a 50×75 instance varies between 7.3 and 27.5 s, and that for a 75×100 instance varies between 77.5 and 1032.2 s.

Regarding the comparison among the heuristics. From Table 4.1, it can be inferred that both the SSr and SSg reach the optimal solution in the 12 instances. Also, it can be seen that GA reaches the optimal solution in 9 of the 12 instances; and in 2 instances obtained the optimal value in the 10 runs, see %A column. However, the SS-GRASP clearly outperforms the other two algorithms in terms of solution quality. Also, SS-GRASP reached the optimal solution for all instances. Furthermore, the average percentage deviation was lower than 0.51% for all small-size instances. Moreover, the time consumed by the SS-GRASP is smaller than SSr and SSg in all instances, which indicates that the hybridization of SS and GRASP is fruitful.

In Table 4.2, the results of the experiments conducted using the 50×75 instances are presented. The computational times of the five algorithms increased as expected, but the increase was not as pronounced as that of the single-level reformulations. In contrast, SS-GRASP maintained the required time under 1 s and reached the optimal value in all 12 instances. Both scatter search approaches keep the good performance. For these three heuristics, the behaviors of %B and %A remain the same as those in the case of the previous small-size instances. In contrast, the quality of the solutions obtained using GA started deteriorating. For example, the GA reached the optimal solution only in 9 of the 12 instances. Furthermore, the results of the 75×100 instances indicate that the SS-GRASP maintains its efficient performance, reaching the optimal value in all instances within a reasonable computational time. On average, the proposed heuristic reaches near optimal solutions, less than 1% in 11 of the 12 instances. The time is significantly reduced with respect of SSr and SSg.

For the large-size instances, the results are summarized in Tables 4.4, 4.5 and 4.6. These tables are similar than in the previous ones. The main difference is that Best Known column, indicates the best value obtained either by a reformulation or a heuristic; and it is used as the reference value used for computing the average deviations. Also, if NEM appears, then it means that the computer ran out of memory while solving the instance. That is, when the message "not enough memory" was given as the output in the optimizer.

Table 4.4: Comparison for the 100×1000 instances.

			P-D.R		MPAC.R		V&K.R		SSr				SSg				SS-GRASP				GA		
Inst	p	Opt	%	t (s)	%	t (s)	%	t (s)	%B	%A	%W	t (s)	%B	%A	%W	t (s)	%B	%A	%W	t (s)	%B	%A	%W
capa_1	7	22696982	95.57	21600.00	52.10	21600.00	52.20	21600.00	0.00	0.22	2.80	68.32	0.00	0.37	1.74	77.34	0.00	0.00	0.00	3.64	4.49	0.42	8.85
capa_2	9	22824578	96.77	21600.00	56.40	21600.00	51.36	21600.00	0.00	0.00	0.00	62.14	0.00	0.72	1.45	74.63	0.00	0.02	0.08	4.19	5.01	0.60	7.10
capa_3	5	22666586	60.98	21600.00	51.19	21600.00	51.19	21600.00	0.00	0.15	1.12	85.87	0.00	0.60	1.12	93.39	0.00	0.03	0.07	3.46	1.44	0.36	4.90
capa_4	7	22760224	68.30	21600.00	53.16	21600.00	56.83	21600.00	0.00	0.00	0.00	78.40	0.00	0.05	0.99	84.56	0.00	0.00	0.00	3.26	2.28	0.69	3.89
capb_1	5	22368690	58.45	21600.00	48.54	21600.00	48.49	21600.00	0.00	0.70	2.59	92.74	0.00	1.52	3.63	96.45	0.00	0.00	0.00	3.27	0.00	0.07	2.45
capb_2	8	22687400	86.80	21600.00	51.04	21600.00	51.04	21600.00	0.57	1.05	2.37	118.90	0.00	0.69	1.28	124.10	0.00	0.00	0.00	3.47	0.00	0.42	2.53
capb_3	5	22307101	57.79	21600.00	48.09	21600.00	48.09	21600.00	0.00	0.00	0.00	97.62	0.00	3.83	8.28	105.40	0.00	0.00	0.00	3.05	0.00	0.17	2.69
capb_4	6	22516656	60.48	21600.00	51.20	21600.00	51.47	21600.00	0.09	3.20	7.48	94.68	0.00	1.02	2.03	98.35	0.00	0.00	0.00	3.27	0.00	0.44	3.59
capc_1	5	21908918	75.05	21600.00	49.11	21600.00	49.11	21600.00	0.00	0.16	2.43	86.91	1.12	2.51	3.89	94.38	0.00	0.00	0.00	2.78	0.00	0.17	2.47
capc_2	10	22186853	88.49	21600.00	56.60	21600.00	56.60	21600.00	0.00	0.03	1.92	108.40	0.34	0.53	0.73	114.90	0.00	0.02	0.08	3.33	3.10	1.62	3.60
capc_3	9	22334094	63.67	21600.00	56.33	21600.00	56.33	21600.00	0.39	2.74	6.75	97.49	0.00	0.26	0.53	109.50	0.00	0.00	0.00	3.56	6.42	0.99	9.62
capc_4	10	22158302	71.44	21600.00	56.05	21600.00	55.86	21600.00	0.24	0.90	4.18	102.40	0.24	0.37	3.94	129.60	0.00	0.00	0.00	4.51	3.51	1.07	3.79
		Average	73.65	21600.00	52.48	21600.00	52.38	21600.00	0.11	0.76	2.64	91.16	0.14	1.04	2.47	100.22	0.00	0.01	0.02	3.48	2.19	0.59	4.62

For 100×1000 instances, the optimizer was not able to optimally solve any problem within the time limit. Moreover, the optimality gap reached by CPLEX was around the 50%

Table 4.5: Comparison for the 300 $\times$ 1000 instances.

			P-D.R		MPAC.R		V&K.R		SSr				SSg				SS-GRASP				GA		
Inst	**p**	**Opt**	**%**	**t (s)**	**%**	**t (s)**	**%**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**
biga_1	22	19009700	-	NEM	-	NEM	-	NEM	0.00	1.59	3.11	221.15	0.07	1.70	8.93	294.28	0.00	0.28	0.75	6.99	3.59	5.27	7.35
biga_2	21	19285300	-	NEM	-	NEM	-	NEM	0.50	2.13	7.27	198.43	0.00	0.94	5.55	301.49	0.00	0.80	1.33	8.01	2.22	3.74	5.66
biga_3	30	19247800	-	NEM	-	NEM	-	NEM	0.00	0.58	1.74	227.33	0.05	2.41	6.47	268.58	0.00	1.15	2.12	7.31	3.93	6.50	8.45
biga_4	24	19760700	-	NEM	-	NEM	-	NEM	1.68	2.65	4.68	204.62	1.23	3.10	5.92	239.18	0.00	1.48	2.27	6.74	3.79	5.20	7.87
bigb_1	25	19203100	-	NEM	-	NEM	-	NEM	0.55	3.43	5.48	227.02	0.00	1.40	6.59	338.22	0.00	0.50	0.91	8.73	4.50	6.10	9.03
bigb_2	27	18791400	-	NEM	-	NEM	-	NEM	0.00	0.32	2.34	250.13	0.00	1.38	4.24	287.70	0.00	0.84	1.69	7.92	8.63	10.59	12.12
bigb_3	19	19428000	-	NEM	-	NEM	-	NEM	0.00	1.17	1.83	183.70	0.00	0.81	3.68	253.49	0.00	0.59	1.20	6.77	0.46	3.40	5.18
bigb_4	22	19774100	-	NEM	-	NEM	-	NEM	0.00	1.97	3.06	180.39	0.00	1.59	3.69	252.78	0.00	0.90	1.67	8.00	3.40	5.42	8.59
bigc_1	21	19631100	-	NEM	-	NEM	88.50	21600	1.76	2.90	4.72	233.24	1.51	3.25	4.59	280.32	0.00	0.78	1.87	8.37	2.63	4.09	5.49
bigc_2	25	18750300	-	NEM	-	NEM	-	NEM	0.82	1.16	3.46	219.23	2.03	3.40	3.75	278.96	0.00	0.55	1.14	6.19	3.53	5.03	7.85
bigc_3	27	18974800	-	NEM	-	NEM	-	NEM	0.36	1.74	2.74	161.93	1.43	4.45	5.92	194.58	0.00	1.10	1.94	7.01	5.15	6.35	9.18
bigc_4	17	19715500	-	NEM	-	NEM	-	NEM	1.51	4.14	5.38	176.05	0.00	0.72	2.37	214.26	0.00	0.56	1.24	7.01	2.00	3.85	5.76
		Average	-	-	-	-	-	-	0.60	1.98	3.82	206.94	0.53	2.10	5.14	266.99	0.00	0.79	1.51	7.42	3.65	5.46	7.71

Table 4.6: Comparison for the 500 $\times$ 1000 instances.

			P-D.R		MPAC.R		V&K.R		SSr				SSg				SS-GRASP				GA		
Inst	**p**	**Opt**	**%**	**t (s)**	**%**	**t (s)**	**%**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**	**t (s)**	**%B**	**%A**	**%W**
largea_1	35	18409200	-	NEM	-	NEM	-	NEM	0.00	0.87	6.91	549.59	0.00	0.16	1.17	611.24	0.00	1.03	2.04	12.20	7.77	9.49	10.91
largea_2	47	17822100	-	NEM	-	NEM	-	NEM	2.45	3.18	4.88	591.44	1.82	2.24	4.77	664.93	0.00	0.83	1.67	12.45	1.58	12.84	13.89
largea_3	49	17380300	-	NEM	-	NEM	-	NEM	0.00	1.72	6.13	637.82	2.15	2.73	3.93	706.67	0.00	0.92	1.70	10.72	5.74	16.03	17.06
largea_4	38	18212500	-	NEM	-	NEM	-	NEM	2.41	2.82	4.75	494.63	0.00	1.74	5.49	543.55	0.00	0.86	1.86	12.98	7.55	11.16	12.14
largeb_1	44	17811000	-	NEM	-	NEM	-	NEM	1.98	2.47	3.91	447.01	0.00	0.95	1.43	514.57	0.00	1.79	2.85	12.98	7.24	16.24	17.76
largeb_2	38	18132800	-	NEM	-	NEM	-	NEM	1.54	1.82	2.37	545.97	0.00	2.38	7.28	627.34	0.00	1.13	2.51	13.18	11.38	13.42	14.38
largeb_3	40	17795700	-	NEM	-	NEM	-	NEM	0.00	0.46	5.36	472.21	1.56	1.87	3.62	554.17	0.00	1.56	3.27	13.45	3.57	12.18	13.36
largeb_4	48	17827900	-	NEM	-	NEM	-	NEM	2.39	2.84	4.94	436.74	1.34	3.32	4.26	526.48	0.00	1.68	2.62	13.72	3.94	12.71	13.95
largec_1	43	17585000	-	NEM	-	NEM	-	NEM	0.00	2.79	3.43	551.11	2.69	3.76	8.73	680.16	0.00	1.20	2.62	13.11	5.87	13.49	14.44
largec_2	36	18314300	-	NEM	-	NEM	-	NEM	1.87	3.65	6.75	460.55	2.05	2.58	2.93	585.95	0.00	1.00	1.83	11.70	8.57	10.54	11.76
largec_3	41	18048400	-	NEM	-	NEM	-	NEM	0.00	2.21	4.78	576.37	0.00	0.52	1.25	688.42	0.00	1.69	2.44	14.23	8.91	12.95	14.78
largec_4	39	17567500	-	NEM	-	NEM	-	NEM	0.00	3.75	5.68	432.08	2.89	3.27	8.91	529.43	0.00	2.05	2.76	11.97	4.65	13.34	14.81
		Average	-	-	-	-	-	-	1.05	2.38	4.99	516.29	1.21	2.13	4.48	602.74	0.00	1.31	2.35	12.72	6.40	12.87	14.10

for MPAC.R and V&K.R. Furthermore, regarding with the 300 $\times$ 1000 instances, the performance of the optimizer get worse. For the P-D.R and MPAC.R reformulations, the memory ran out for all the 12 instances and in 11 for the V&K.R reformulation. However, for the other instance the gap reached was of 88.5%. As it was expected, the results for the 500 $\times$ 1000 instances do not show any significant information; the NEM status appeared in all the reformulations for the 12 instances. Based on these results, it is evident that the reformulations are not able to solve large-size instances and the use of heuristics algorithms is justified.

Regarding the heuristics, Table 4.4 shows that the SSr, SSg, and GA reached the best value obtained by the SS-GRASP only in 8, 9, and 5 of the 12 instances, respectively. Also, their percentage deviations of the best value are lower than 3.20%, 3.83%, and 6.42%. As it is described above, the SS-GRASP reached the best known solution in all the instances. Furthermore, it obtained the best known solution in the 10 runs for 9 of the 12 instances, see %A column. The latter clearly indicates the efficient performance of this proposed algorithm.

Finally, Tables 4.5 and 4.6 evidence an increasing in the required time due to the increase in the size of the instances. We can see that the GA not reached the best known value obtained by the SS-GRASP in any of the 24 instances. On the contrary, SSr and SSr reach the best solution in 11 of the 24 instances, each of them. It can be inferred from the %B columns that the worst values reached by the SSr are 1.76% (Table 4.5) and 2.45% (Table 4.6); by the SSg are 2.03% and 2.89%; and by the GA are 8.63% and 11.38%, respectively. On the other hand, for the percentage deviation for the average values, the SS-GRASP shows a 2.05% as the greater gap obtained in these 24 instances, which is an acceptable value taking into account the size of the instances.

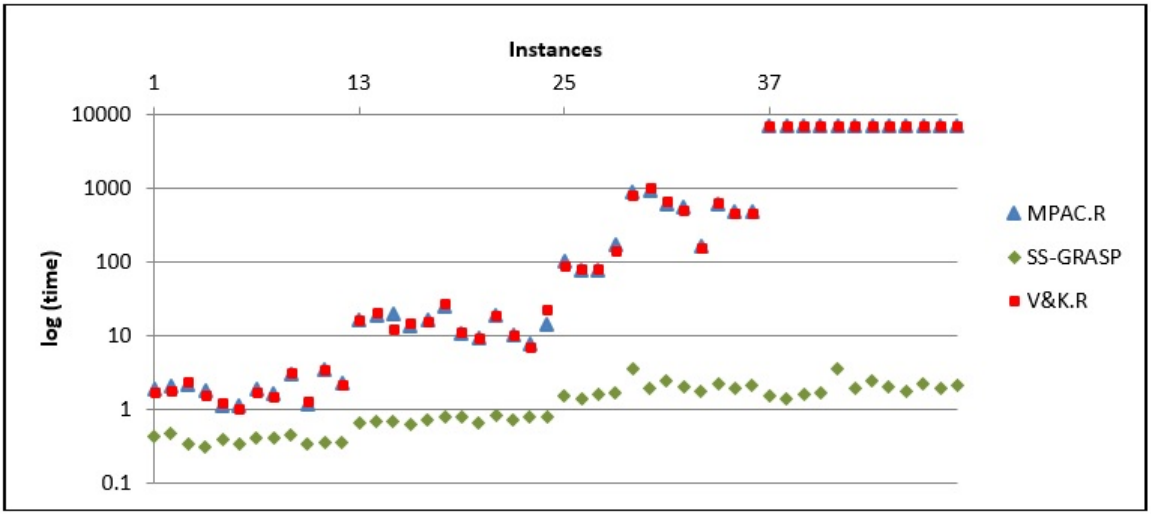
It is worthy to mention that despite the fact the scatter search algorithms (SSr and SSg)

show an acceptable performance in terms of solution quality; the computational time dramatically augments as the size of instance increases. The latter could be attached to the fact that a full exploration of the neighborhood considered in the local search is made. Hence, the reduction in the neighborhood size positively impacts the performance of the SS-GRASP. Moreover, from the last row of Tables 4.1-4.6 it can be seen that the average of percentage deviations is maintained for all the instances regardless the size of the instance. This is a good attribute for the proposed hybrid heuristic.

4.1.4.2 Comparing the required computational time

After discussing the efficiency of the hybrid algorithm for solving p-median BPO, we compared the required time for optimally solving both integer single-level reformulations MPAC.R and V&K.R. Figure 4.4 plots the required time for solving the first 48 instances reported for the MPAC.R, V&K.R, and the proposed heuristic. It is evident that plotting the time reported in the P-D.R column or the two larger set of instances will not give any relevant information. Because the whole set of instances is divided into six equal different subsets, the plotted results appear in a grouped manner (four groups in this case). The first 12 data entries correspond to the 50×50 instances, entries 1324 correspond to the 50×75 instances, the following entries correspond to the 75×100 and the final 12 entries are associated with the 100×1000 instances. To display the results properly, the data were plotted on a logarithmic scale. The reason for scaling one axis is the magnitude of the reported values of the 75×100 and 100×1000 instances solved using CPLEX. In Figure 4.4, it can be seen that although the required time of the SS-GRASP increases, the increase is a small polynomial one, whereas the increase corresponding to the single-level reformulations seems to be exponential (in the cases that the instances were solved).

Figure 4.4: Required computational time for solving instances.



Chapter 5

Covering Problems

5.1 A Bilevel Maximal Covering Location Problem

A bilevel model for the problem herein studied is presented. In addition, two equivalent single-level reformulations are studied. Also, a genetic algorithm (GA) is proposed to obtain good quality lower bounds for the problem, where the objective function value is computed following the ideas described in [92] and [22]. The proposed algorithm is tested with a set of randomly generated instances. The parameter tuning conducted for the genetic algorithm is detailed in a didactical manner. In order to evaluate the behavior of the proposed genetic algorithm, the obtained solutions are compared with the lower bounds obtained with one of the single-level reformulations, within a three-hour time limit. According to this comparison, the proposed genetic algorithm provides very good quality solutions with small computational effort.

Later, analyzing the previous results we decided to improve the quality of the solutions and the required computational time. For this, another equivalent single-level reformulation, a greedy randomized adaptive search procedure (GRASP) heuristic and a hybrid GRASP-Tabu heuristic are proposed. The hybrid GRASP-Tabu heuristic combines GRASP and tabu search to efficiently find lower bounds for large-scale instances for the maximal covering location problem with customer preference ordering. According to computational results, despite their simplicity, the proposed algorithms provide optimal or near optimal solutions in a small computation time.

The remainder of this chapter is as follows. Section 5.1.1 presents the problem's description and the proposed mathematical bilevel model. In Section 5.1.2, two reformulations of the bilevel problem that reduce it into a single-level one are described. Section 5.1.3 details the description of the genetic algorithm developed to obtain good quality solutions of the bilevel problem. The computational experimentation conducted to assess the performance of the proposed genetic algorithm is shown in Section 5.1.5. The third proposed reformulation is presented in 5.1.6. In section 5.1.7, the GRASP and the hybrid GRASP-Tabu algorithms are described. Finally, computational experiments regarding the single-level reformulation, GRASP and GRASP-Tabu are reported in 5.1.8.

5.1.1 Problem statement and mathematical model

In this section, the problem and a mathematical formulation of the maximal covering location bilevel problem (MCLBP) are described. The problem studied considers a firm that wants to enter a market where a set of firms already exist. The firm wants to locate p facilities to maximize captured demand. We assume that customers have the freedom to patronize the facility of their preference within a predefined distance threshold. This situation can be formulated as a bilevel programming problem, where the upper level is associated with the new competitor and the lower level corresponds to the customers. The upper level decides the location of a limited number of facilities, and the lower level allocates customers to their most preferred facility. It is also assumed, that existing facilities will remain opened as they belong to other firms.

Notation

Now, let us properly define the sets, parameters and decision variables involved in the proposed formulation for the MCLBP.

Sets

I_1	Set of potential facilities
I_2	Set of existing facilities
J_1	Set of customers uncovered by I_2
J_2	Set of customers covered by I_2
$I(j)$	Patronizing set of customer $j \in J$ (i.e. the set of facilities that cover customer j)
$J(i)$	Patronizing set of location $i \in I$ (i.e. the set of customers that are covered by location i)

where $I = I_1 \cup I_2$, and $J = J_1 \cup J_2$.

Parameters

D_j	Demand associated with customer $j \in J$
p	Number of facilities that the leader will open for entering in the market
g_{ij}	Preference of the customer $j \in J$ towards the facility $i \in I$
d_{ij}	Distance from customer $j \in J$ to facility $i \in I$
r	Predetermined coverage radius

Binary decision variables

Leader	y_i	binary variable which indicates whether the facility $i \in I_1$ is opened or not.
Follower	x_{ij}	binary variable which indicates whether the customer $j \in J$ is allocated to the facility $i \in I$ or not.

Finally, it is convenient to mention that customer preferences are assumed to be consecutive integer numbers from 1 to $|I(j)|$. Also, for identifying sets $I(j)$ and $J(i)$ the following inequality must be considered, $d_{ij} \leq r, i \in I, j \in J$.

The proposed mathematical model for the MCLBP is as follows:

$$\max_{y,x} \quad \sum_{i\in I_1}\sum_{j\in J(i)} D_j x_{ij} \tag{5.1}$$

$$\text{subject to:} \quad y_i = 1 \qquad \forall i \in I_2 \tag{5.2}$$

$$\sum_{i\in I_1} y_i = p \tag{5.3}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{5.4}$$

$$\text{where} \quad x \quad \text{solves}$$

$$\max_{x} \quad \sum_{i\in I}\sum_{j\in J(i)} g_{ij} x_{ij} \tag{5.5}$$

$$\text{subject to:} \quad \sum_{i\in I(j)} x_{ij} = 1 \qquad \forall j \in J_1 \tag{5.6}$$

$$x_{ij} \leq y_i \qquad \forall i \in I_1, j \in J(i) \tag{5.7}$$

$$\sum_{i\in I(j)} x_{ij} \leq 1 \qquad \forall j \in J_2 \tag{5.8}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J(i) \tag{5.9}$$

Expression (5.1) is the leader objective function which maximizes demand covered by open facilities. Constraints (5.2) ensure that existing facilities will remain open. Constraints (5.3) guarantee that exactly p facility locations will be selected by the firm. The follower's objective function is given by (5.5) which maximizes customer preferences. Constraints (5.6) guarantee that customers already covered by an existing facility will remain covered by an open facility. Constraints (5.7) ensure that demand cannot be allocated to locations where no facility exists. These constraints also ensure that if the demand of customer j is allocated to facility i, this facility belongs to the patronizing set of customer j (i.e. $i \in I(j)$). Constraints (5.8) guarantee that the demand not covered by the existing facilities can be covered by at most one open facility or remain uncovered. Finally, the binary nature of the leader's and follower's decision variables is specified in (5.4) and (5.9).

In order to have a well-defined bilevel problem it is important to carefully analyze the existence and uniqueness of the lower level optimal solution. In other words, if the follower's problem has a non-unique optimal solution for any leader's decision, then the bilevel problem is ill-posed. Fortunately, in our case, the lower level has a unique optimal solution for any leader's decision since customer preferences are positive consecutive numbers from 1 to $|I|$. The proof of this important result is given in [136].

5.1.2 Model reformulations

The MCLBP described in the previous section will be reduced into a single-level mixed integer programming problem. The reformulation is made following the classical approach of using the primal-dual relationships for the follower's problem when follower's problem satisfies strong duality. To ensure optimality for the follower's problem, two schemes are commonly considered: (i) force the equality of the objective functions of both primal and dual problems

and (ii) include the complementarity slackness constraints. In this section, both schemes are presented.

For a given fixed leader's decision vector $\hat{y}$ the follower's problem can be seen as a parametrized optimization problem. The LP relaxation of the resulting problem is:

$$\max \quad \sum_{i\in I}\sum_{j\in J(i)} g_{ij}x_{ij} \tag{5.10}$$

$$\text{subject to:} \quad \sum_{i\in I(j)} x_{ij} = 1 \qquad \forall j \in J_2 \tag{5.11}$$

$$\sum_{i\in I(j)} x_{ij} \leq 1 \qquad \forall j \in J_1 \tag{5.12}$$

$$0 \leq x_{ij} \leq \hat{y}_i \qquad \forall i \in I, j \in J(i) \tag{5.13}$$

The optimal solution of the LP relaxation of the lower level problem can be easily computed in the following manner. If $\hat{y}_i = 0$, then $x_{ij} = 0, \forall j \in J$. Let I' be the index set of the facility locations selected by the leader, $I' = \{i \in I_1 : \hat{y}_i = 1\}$, and let $I^\star = I' \cup I_2$ be the set of open facilities. Then, the demand of each customer is allocated to the most preferred facility location within the open facilities of the patronizing set of that customer, that is, customer j is allocated to the open facility $i(j) = \arg\max_{i\in I(j)\cap I^\star}\{g_{ij}\}$ whenever $I(j)\cap I^\star \neq \emptyset$, otherwise, the demand of customer j is not covered by open facilities. Therefore, the optimal solution for the LP relaxation is as follows: $x_{i(j),j} = 1, \forall i \in I, j \in J$ such that $I(j) \cap I^\star \neq \emptyset$, and $x_{ij} = 0, \forall i \in I \setminus I^\star, j \in J$, and $\forall i \in I, j \in J$ such that $I(j) \cap I^\star = \emptyset$

As can be observed, the optimal solution of the LP relaxation of the follower's problem is always an integer solution, and therefore, the lower level problem satisfies strong duality.

In order to reformulate the bilevel problem by any of both approaches described above, the dual problem associated with the follower's problem will be needed.

Let α_j for all $j \in J_2$, β_{ij} for all $i \in I_1, j \in J(i)$ and γ_j for all $j \in J_1$ be the dual variables associated with the linear relaxation of the follower's problem defined by (5.5)- (5.9). Then, the corresponding dual problem is as follows:

$$\max_{\alpha,\beta,\gamma} \quad \sum_{j\in J_1}\alpha_j + \sum_{i\in I_1}\sum_{j\in J(i)} y_i\beta_{ij} + \sum_{j\in J_2}\gamma_j \tag{5.14}$$

$$\text{subject to:} \quad \alpha_j + \beta_{ij} \geq g_{ij} \qquad \forall j \in J_1, i \in I_1 \cap I(j) \tag{5.15}$$

$$\beta_{ij} + \gamma_j \geq g_{ij} \qquad \forall j \in J_2, i \in I_1 \cap I(j) \tag{5.16}$$

$$\alpha_j \geq g_{ij} \qquad \forall j \in J_1, i \in I_2 \cap I(j) \tag{5.17}$$

$$\gamma_j \geq g_{ij} \qquad \forall j \in J_2, i \in I_2 \cap I(j) \tag{5.18}$$

$$\beta_{ij} \geq 0 \qquad \forall i \in I_1, j \in J(i) \tag{5.19}$$

$$\gamma_j \geq 0 \qquad \forall j \in J_2 \tag{5.20}$$

The first approach considered for reformulating the bilevel problem consists in adding the constraint that indicates the equality of the objective functions of the linear relaxation of the

follower's problem and its dual. It is easy to see that the second term in the dual's objective function is non-linear. Hence, an auxiliary variable is introduced for linearizing it. Let $\delta_{ij} = y_i\beta_{ij}$ for all $i \in I_1, j \in J(i)$. It is worthy to note that, since y_i only can take the values 0 or 1, then $\delta_{ij} = \beta_{ij}$ when $y_i = 1$ and $\delta_{ij} = 0$ when $y_i = 0$. Therefore, the following constraints are added:

$$\begin{aligned}
\delta_{ij} &\geq 0 \qquad \forall i \in I_1, j \in J(i) && (5.21)\\
\delta_{ij} &\leq My_i \qquad \forall i \in I_1, j \in J(i) && (5.22)\\
\delta_{ij} &\leq \beta_{ij} \qquad \forall i \in I_1, j \in J(i) && (5.23)\\
\delta_{ij} + M &\geq \beta_{ij} + My_i \qquad \forall i \in I_1, j \in J(i) && (5.24)
\end{aligned}$$

where M is a sufficiently large positive constant.

As a result, the resulting single-level mixed integer programming problem is as follows:

$$\max_{y,x,\alpha,\beta,\gamma,\delta} \quad \sum_{i\in I_1}\sum_{j\in J(i)} D_j x_{ij} \qquad (5.25)$$

$$\begin{aligned}
\text{subject to:} \quad y_i &= 1 \qquad \forall i \in I_2 && (5.26)\\
\sum_{i\in I_1} y_i &= p && (5.27)\\
\sum_{i\in I(j)} x_{ij} &= 1 \qquad \forall j \in J_1 && (5.28)\\
x_{ij} &\leq y_i \qquad \forall i \in I_1, j \in J(i) && (5.29)\\
\sum_{i\in I(j)} x_{ij} &\leq 1 \qquad \forall j \in J_2 && (5.30)\\
\alpha_j + \beta_{ij} &\geq g_{ij} \qquad \forall j \in J_1, i \in I_1 \cap I(j) && (5.31)\\
\beta_{ij} + \gamma_j &\geq g_{ij} \qquad \forall j \in J_2, i \in I_1 \cap I(j) && (5.32)\\
\alpha_j &\geq g_{ij} \qquad \forall j \in J_1, i \in I_2 \cap I(j) && (5.33)\\
\gamma_j &\geq g_{ij} \qquad \forall j \in J_2, i \in I_2 \cap I(j) && (5.34)\\
\sum_{i\in I}\sum_{j\in J(i)} g_{ij}x_{ij} &= \sum_{j\in J_1}\alpha_j + \sum_{i\in I_1}\sum_{j\in J(i)}\delta_{ij} + \sum_{j\in J_2}\gamma_j && (5.35)\\
\delta_{ij} &\geq 0 \qquad \forall i \in I_1, j \in J(i) && (5.36)\\
\delta_{ij} &\leq My_i \qquad \forall i \in I_1, j \in J(i) && (5.37)\\
\delta_{ij} &\leq \beta_{ij} \qquad \forall i \in I_1, j \in J(i) && (5.38)\\
\delta_{ij} + M &\geq \beta_{ij} + My_i \qquad \forall i \in I_1, j \in J(i) && (5.39)\\
y_i &\in \{0,1\} \qquad \forall i \in I && (5.40)\\
x_{ij} &\geq 0 \qquad \forall i \in I, j \in J(i) && (5.41)\\
\beta_{ij} &\geq 0 \qquad \forall i \in I_1, j \in J(i) && (5.42)\\
\gamma_j &\geq 0 \qquad \forall j \in J_2 && (5.43)
\end{aligned}$$

The second approach followed is based on the use of the optimality slackness constraints associated to both primal and dual follower's problems. These constraints substitute the equality

of the respective follower's objective functions. The constraints included are:

$$
\begin{aligned}
(\alpha_j + \beta_{ij} - g_{ij})x_{ij} &= 0 \qquad \forall j \in J_1, i \in I_1 \cap I(j) && (5.44)\\
(\beta_{ij} + \gamma_j - g_{ij})x_{ij} &= 0 \qquad \forall j \in J_2, i \in I_1 \cap I(j) && (5.45)\\
(\alpha_j - g_{ij})x_{ij} &= 0 \qquad \forall j \in J_1, i \in I_2 \cap I(j) && (5.46)\\
(\gamma_j - g_{ij})x_{ij} &= 0 \qquad \forall j \in J_2, i \in I_2 \cap I(j) && (5.47)\\
(y_i - x_{ij})\beta_{ij} &= 0 \qquad \forall J \in J, i \in I_1 \cap I(j) && (5.48)\\
\left(1 - \sum_{i \in I(j)} x_{ij}\right)\gamma_j &= 0 \qquad \forall j \in J_2 && (5.49)
\end{aligned}
$$

It is evident that constraints (5.44)-(5.49) are not linear. Bearing in mind that in the complementarity equations (5.44)-(5.47), the term x_{ij} is binary; in (5.48), the term $y_i - x_{ij}$ can only take 0 or 1 values; and since $\sum_{i \in I(j)} x_{ij} \leq 1, \forall j \in J_1$, then $\left(\sum_{i \in I(j)} x_{ij}\right)$ in equation (5.49) will be equal to 0 or 1, the set of constraints (5.44)-(5.49) can be substituted by the following constraints:

$$
\begin{aligned}
\alpha_j + \beta_{ij} - g_{ij} &\leq M(1 - x_{ij}) \qquad \forall j \in J_1, i \in I_1 \cap I(j) && (5.50)\\
\beta_{ij} + \gamma_j - g_{ij} &\leq M(1 - x_{ij}) \qquad \forall j \in J_2, i \in I_1 \cap I(j) && (5.51)\\
\alpha_j - g_{ij} &\leq M(1 - x_{ij}) \qquad \forall j \in J_1, i \in I_2 \cap I(j) && (5.52)\\
\gamma_j - g_{ij} &\leq M(1 - x_{ij}) \qquad \forall j \in J_2, i \in I_2 \cap I(j) && (5.53)\\
\beta_{ij} &\leq M(1 - y_i + x_{ij}) \qquad \forall j \in J, i \in I_1 \cap I(j) && (5.54)\\
\gamma_j &\leq M\left(\sum_{i \in I(j)} x_{ij}\right) \qquad \forall j \in J_2 && (5.55)
\end{aligned}
$$

where, as before, M is a sufficiently large constant.

Therefore, the second reformulation from is the following single-level mixed integer linear programming problem:

$$\max_{y,x,\alpha,\beta,\gamma} \quad \sum_{i\in I_1}\sum_{j\in J(i)} D_j x_{ij} \tag{5.56}$$

$$\text{subject to:} \quad y_i = 1 \qquad \forall i \in I_2 \tag{5.57}$$

$$\sum_{i\in I_1} y_i = p \tag{5.58}$$

$$\sum_{i\in I(j)} x_{ij} = 1 \qquad \forall j \in J_1 \tag{5.59}$$

$$x_{ij} \leq y_i \qquad \forall i \in I_1, j \in J(i) \tag{5.60}$$

$$\sum_{i\in I(j)} x_{ij} \leq 1 \qquad \forall j \in J_2 \tag{5.61}$$

$$\alpha_j + \beta_{ij} \geq g_{ij} \qquad \forall j \in J_1, i \in I_1 \cap I(j) \tag{5.62}$$

$$\beta_{ij} + \gamma_j \geq g_{ij} \qquad \forall j \in J_2, i \in I_1 \cap I(j) \tag{5.63}$$

$$\alpha_j \geq g_{ij} \qquad \forall j \in J_1, i \in I_2 \cap I(j) \tag{5.64}$$

$$\gamma_j \geq g_{ij} \qquad \forall j \in J_2, i \in I_2 \cap I(j) \tag{5.65}$$

$$\alpha_j + \beta_{ij} - g_{ij} \leq M(1 - x_{ij}) \qquad \forall j \in J_1, i \in I_1 \cap I(j) \tag{5.66}$$

$$\beta_{ij} + \gamma_j - g_{ij} \leq M(1 - x_{ij}) \qquad \forall j \in J_2, i \in I_1 \cap I(j) \tag{5.67}$$

$$\alpha_j - g_{ij} \leq M(1 - x_{ij}) \qquad \forall j \in J_1, i \in I_2 \cap I(j) \tag{5.68}$$

$$\gamma_j - g_{ij} \leq M(1 - x_{ij}) \qquad \forall j \in J_2, i \in I_2 \cap I(j) \tag{5.69}$$

$$\beta_{ij} \leq M(1 - y_i + x_{ij}) \qquad \forall j \in J, i \in I_1 \cap I(j) \tag{5.70}$$

$$\gamma_j \leq M\left(\sum_{i\in I(j)} x_{ij}\right) \qquad \forall j \in J_2 \tag{5.71}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{5.72}$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J(i) \tag{5.73}$$

$$\beta_{ij} \geq 0 \qquad \forall i \in I_1, j \in J(i) \tag{5.74}$$

$$\gamma_j \geq 0 \qquad \forall j \in J_2 \tag{5.75}$$

Since no commercial software exists for solving general bilevel programming problems, these single-level reformulations can be used for solving the problem. Unfortunately, only limited size instances (up to 1000 customers and 100 facilities) can be optimally solved by the proposed reformulations within a reasonable amount of computer effort. Due to this fact, a heuristic method is proposed for obtaining lower bounds for the MCLBP. The next section describes the genetic algorithm proposed for solving the bilevel model defined in (5.1)-(5.9).

5.1.3 Genetic algorithm

Genetic algorithms have been used to find feasible bounds for a wide variety of applications, and they have been a very powerful technique for handling complex problems. In this study,

we propose a genetic algorithm (GA) to find lower bounds for MCLBP. GAs have been applied for many bilevel programming problems with good results. For example, in [96], [68], [20] GAs were used to solve linear bilevel problems; in [102] GAs were used to solve linear and non-linear bilevel problems, in [83] a GA was used to solve a special class of non-linear bilevel programming problem and in [138] a GA was used to solve a quadratic bilevel problem. In addition, [16] proposed a GA for a competitive facility location problem, and [7] used a GA to analyze the vulnerability of a power system. Recently, [23] proposed a GA to obtain good quality configurations for the topological design of LANs. Now, we describe the GA proposed in this study.

To a large extent, the efficiency of genetic algorithms depends on correctly encoding a problem's solution. In this study, leader's solutions are coded with a vector of size p in which each position indicates the index of an opened facility; indices are ordered in an increasing manner. The follower's solution is coded with a vector of size $|J|$ (i.e. the total number of customers). In this vector, the value of the j^{th} position corresponds to the facility to which customer j^{th} is allocated (a zero value means that customer j is not covered by any open facility). For example, consider a problem with 8 customers, 10 potential facilities and $p = 5$. Also, let $[2, 5, 6, 8, 9]$ be a solution for the leader's problem, and let $[5, \underline{11}, 6, 2, 9, \underline{3}, 5, 8]$ be a solution for the follower's problem. We observe that customers 1 and 7 are allocated to facility 5, customer 3 is allocated to facility 6, customer 4 is allocated to facility 2, customer 5 is allocated to facility 9, customer 8 is allocated to facility 8 and customers 2 and 6 are allocated to existing facilities 11 and 3, respectively. It is worth noting that facilities 3 and 11 belong to competitors and they are the most preferred facility for customers 6 and 2, respectively.

It is important to highlight that the solutions considered in our GA are the ones associated with the leader's decision. For each leader's solution, the follower's problem is solved to optimality. The optimal solution of the lower level can be found efficiently, by rearranging the preferences matrix in the same way as in [92] and [22], but considering the covering constraint.

Initial population

To generate the initial population, a biased random methodology is used. The locations are randomly selected from the set of non-opened potential locations until p locations are opened. The probability to open facility i is computed in the following manner:

$$q_i = \frac{\sum_{j \in J(i)} b_j}{\sum_{i \in I_1} \sum_{j \in J(i)} b_j} \tag{5.76}$$

for all $i \in I_1$, in order to favor those locations that might cover more demand.

Selection mechanism

One of the most common methods for avoiding premature convergence in the selection phase is the tournament strategy. This strategy consists in the following: for each solution of the

population, five tournaments are made against another solution randomly selected from the population (as in [22]). In each tournament the leader's objective function values (fitness) are compared. The number of times that each solution wins a tournament is recorded. After the tournaments are performed, the solutions of the populations are sorted in non-increasing order with respect to the number of times that each solution wins the tournaments. An elitist selection is made to prevail the good quality solutions.

Genetic operators

Genetic operators are used to obtain new solutions to explore the solutions' space. The aim is to reach better solutions starting from the current ones. Two different operators are considered: crossover and mutation. Crossover operator generates two solutions (offspring) that will inherit some characteristics from the two selected initial solutions (parents). The mutation operator generates a new solution (offspring) that is not very different from the selected initial solution.

Now we describe the crossover operator used in the proposed GA. Two solutions (chromosomes) are selected from the current population. A random integer number between 1 and $p - 1$ that is used to define the position in which both chromosomes will be divided is generated. The two parent chromosomes are combined using the first part of one of the parents and the second part of the other to generate two offspring. It is evident that the heritability principle for crossover in genetic algorithms is maintained, due to the fact that the offspring will have similar characteristics than the parents. However, the offspring not always represent feasible solutions since it might be possible that they contain the same opened facility in two different positions. In this case, the infeasible chromosome is eliminated. Finally, repeated solutions are eliminated from the population.

The aim of the mutation operator is to incorporate diversity in the current population. This procedure ensures that the resulting solution will be different from the original one. The mutation operation is applied to a single solution at a time in the following way: a position is selected randomly and the facility in that position is replaced by a non-open facility randomly selected.

The genetic algorithm used in this chapter can be summarized in the following way. An initial population is generated. In order to apply the genetic operators to the best solutions, the tournament procedure is executed. Then, the best half of the population is used to apply the genetic operators. For each of these solutions, the genetic operator to be applied is randomly selected. If the crossover is going to be applied the second parent is randomly selected from the best half of the population. Otherwise, the mutation operator is applied to that solution. The population is updated in an elitist way based on tournaments results. The stopping criterion is a fixed number of generations. It is convenient to highlight that after a new leader's solution is obtained, the lower level problem is optimally solved and then the solution fitness value is computed.

5.1.4 Preliminary tests and parameter tuning

A set of 60 randomly generated instances was used in this study for testing the proposed algorithm. We generated the random instances using a methodology based on the procedure described in [113]. First, t points were generated in a unit square and the corresponding matrix of Euclidean distances was computed. Then, m of those points were randomly selected to be facility locations, and the other $n = t - m$ points were set as customers. Then, the set of m facilities was randomly divided into two sets. We selected 10% of the facilities to be the existing ones, and the remaining 90% were set as the potential facility locations. Finally, customer preferences towards facilities were generated using the process described in [26]. Test instances were partitioned in six subsets according to their size. It is convenient to note that for each size, a specific coverage ratio r was selected. The first three subsets are considered medium size (block M); while, the others are referred as large size instances (block L). The description of the sizes is shown in Table 5.1. The coverage ratio r of subsets 1 to 6 were set to 0.80, 0.70, 0.50, 0.30, 0.25 and 0.20, respectively.

It is required to determine the value of three parameters for the proposed genetic algorithm: population size, number of generations and probability of selecting the crossover operator. To select the most appropriate parameter values, preliminary tests with different values for each parameter were conducted. Four different population sizes (P_{size}) were considered: 100, 200, 500, and 1000 individuals. For the number of generations, two values proportional to the total number of facilities were considered: $2|I|$, and $3|I|$. Four values for the crossover probability were used: 0.65, 0.75, 0.85, and 0.95 (therefore, mutation probabilities were 0.35, 0.25, 0.15, and 0.05, respectively).

To conduct a full factorial design, the algorithm would need to be executed (4)(2)(4)=32 times for each instance. Therefore, to avoid an excessive computational effort, a subset of 30 instances (block M) was considered for running all the 32 possible parameter configurations. After a preliminary test, we noticed that the behavior of the algorithm was very similar for same size instances. Thus, to adjust the algorithm parameters, we selected three instances of each size of block M.

Table 5.1: Instances characteristics.

	Block M			***Block L***		
Customers	225	450	675	900	1350	1800
Facilities	25	50	75	100	150	200

For each instance, the algorithm was executed five times for each of the 32 parameter configurations. The following values of the leader' objective function were recorded: the best value out of the five executions and its average. Also, the average CPU time of the five executions was recorded. For all configurations, the algorithm obtained the optimum or the best known solution, for every instance, in at least one out of the five runs (this was validated comparing the obtained value against the value obtained by one of the reformulations described previously).

Table 5.2: Percentage deviations between the optimum and the GA with P_{size}=*100*.

Configuration	*225×25-01*	*225×25-08*	*225×25-10*	*450×50-13*	*450×50-16*	*450×50-18*	*675×75-23*	*675×75-25*	*675×75-29*
(2|I|,0.65)	0.00	0.00	0.00	0.05	0.36	1.16	0.64	0.38	0.84
(2|I|,0.75)	0.00	0.00	0.00	0.04	0.33	0.31	1.06	0.17	0.56
(2|I|,0.85)	0.00	0.00	0.00	0.15	0.00	0.43	0.05	0.30	0.31
(2|I|,0.95)	0.00	0.00	0.00	0.04	0.24	0.40	0.07	0.12	0.53
(3|I|,0.65)	0.00	0.00	0.00	0.05	0.24	0.42	0.59	0.61	0.88
(3|I|,0.75)	0.00	0.00	0.00	0.03	0.45	0.55	0.60	0.13	0.84
(3|I|,0.85)	0.00	0.00	0.00	0.05	0.24	0.33	0.32	0.48	0.69
(3|I|,0.95)	0.00	0.00	0.00	0.39	0.00	0.00	0.18	0.12	0.66

Table 5.3: Percentage deviations between the optimum and the GA with P_{size}=*200*.

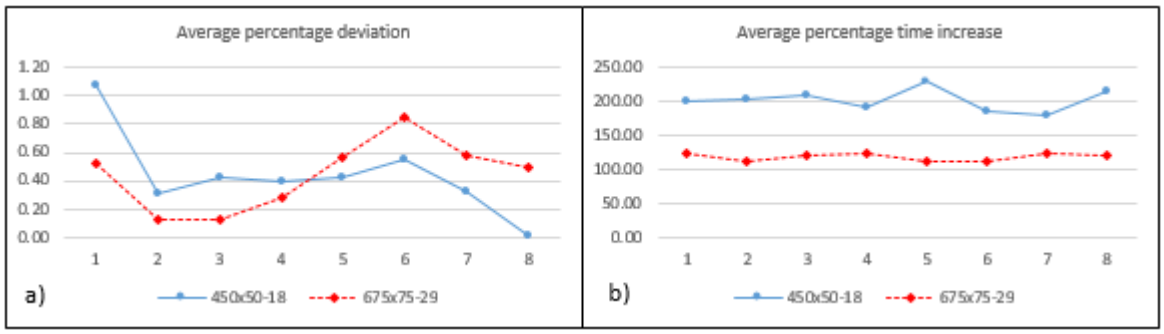
Configuration	*225×25-01*	*225×25-08*	*225×25-10*	*450×50-13*	*450×50-16*	*450×50-18*	*675×75-23*	*675×75-25*	*675×75-29*
(2|I|,0.65)	0.00	0.00	0.00	0.00	0.00	0.09	0.03	0.07	0.32
(2|I|,0.75)	0.00	0.00	0.00	0.04	0.00	0.00	0.18	0.00	0.43
(2|I|,0.85)	0.00	0.00	0.00	0.03	0.00	0.00	0.17	0.00	0.18
(2|I|,0.95)	0.00	0.00	0.00	0.01	0.00	0.00	0.14	0.00	0.25
(3|I|,0.65)	0.00	0.00	0.00	0.01	0.00	0.00	0.12	0.05	0.32
(3|I|,0.75)	0.00	0.00	0.00	0.02	0.08	0.00	0.08	0.21	0.00
(3|I|,0.85)	0.00	0.00	0.00	0.02	0.00	0.00	0.12	0.15	0.12
(3|I|,0.95)	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.18

Analyzing the results, it was observed that solution quality improved when population size was increased from $P_{size} = 100$ to $P_{size} = 200$, and from $P_{size} = 200$ to $P_{size} = 500$. But, the solution quality did not improved when P_{size} was increased from 500 to 1000. For the smaller instances, the optimal solution was obtained regardless the size of the population. It was also observed, as it was expected, that the CPU time was increased in a significant way as the population size grows. For example, when $P_{size} = 500$ or $P_{size} = 1000$, the algorithm obtained the optimum or the best known objective values in the five runs, but the CPU time increased drastically (more than 10 times with respect to $P_{size} = 100$). Then, we compared solution quality and CPU time for $P_{size} = 200$ and $P_{size} = 100$. Tables 5.2 and 5.3 show the results for the sampled instances and each parameter configuration. We observed that the quality of the solutions obtained for both population sizes (100 and 200) was very similar. Additionally, increasing population size from 100 to 200, also increased significantly the CPU time.

Figure 5.1 exemplifies the impact in the algorithm's performance when increasing the population size from 100 to 200 for two particular instances. For this comparison, we selected the two instances with the highest deviation with respect to the optimum or best known solution. The figure shows the average percentage deviation of the leader's objective function with respect to the optimum or best known solution (Figure 5.1.a), and the average percentage time increased when the population size is increased (Figure 5.1.b). In both graphs, the horizontal axis indicates each parameter configuration. We observed that if population size was reduced from 200 to 100, in the worst case, solution quality worsens only 1.063 %. However, increasing population size from 100 to 200 incremented CPU time to more than double. Therefore, the improvement obtained in the objective function value did not justify the increase in population size. Thus, population size was set to 100, since the worst percent deviation with respect to the optimum or best known objective function value was 1.16 %, which is acceptable for a problem of this nature.

To adjust the parameters associated with the number of generations and the probability for selecting the genetic operators, we used only the test results for the case when P_{size}=100. To

Figure 5.1: Impact from increasing the population size.



select those values, we aimed to determine which configuration dominates the others. Figure 5.2 shows the objective function values against execution times for every sampled instance. The eight possible configurations are plotted for each sampled instance. The label in each point indicates the corresponding configuration (the order is the same as in the previous tables). In this analysis, if two configurations have the same value for one criterion, the one that has the worst value in the other criterion is considered as the dominated solution.

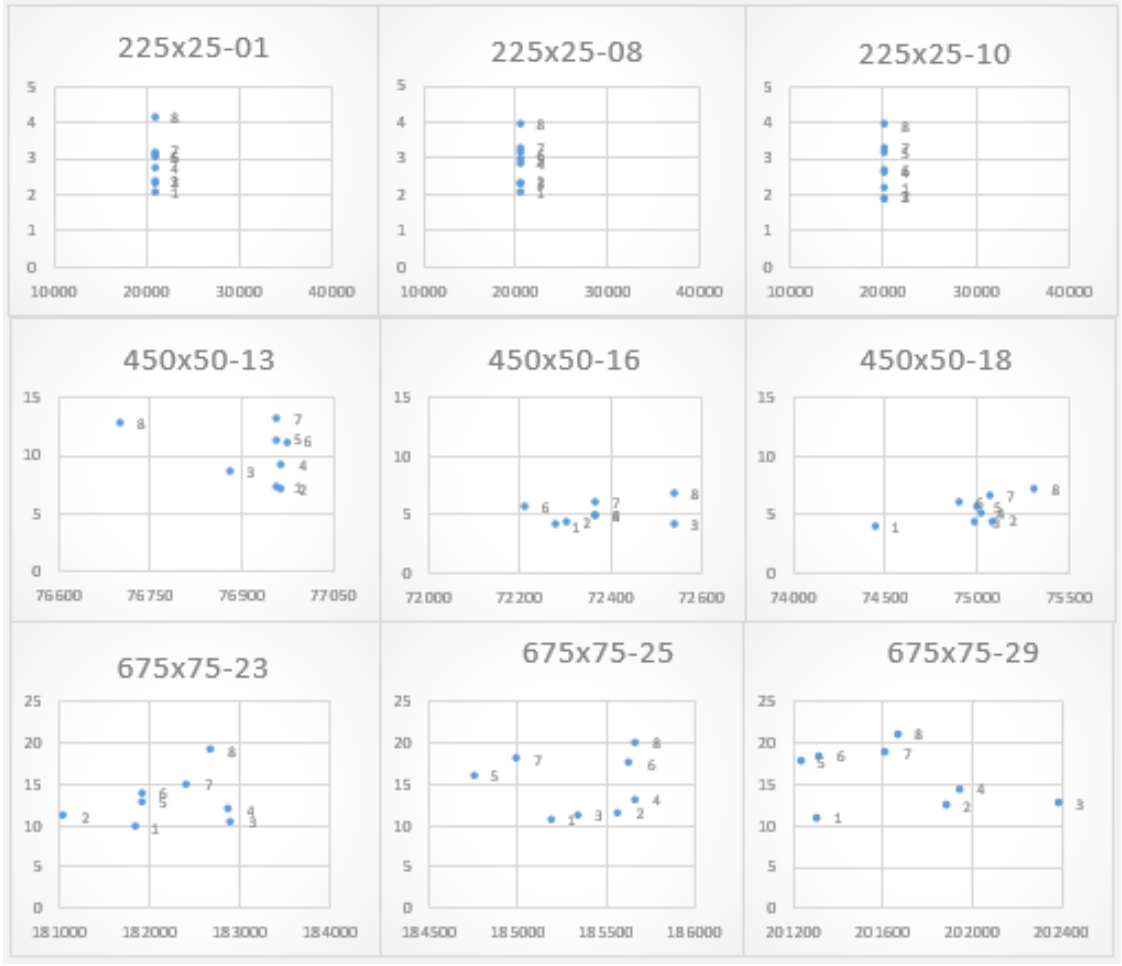
Analyzing Figure 5.2, it can be appreciated that there are some configurations that are non-dominated in most of the sampled instances. For example, configuration $(2|I|, 0.65)$ -label 1- is non-dominated because it is the one with the lowest required CPU time. This is because the probability of selecting the mutation operator is high, which implies that fewer solutions are explored because with mutation operator only one offspring is obtained, while with crossover two offspring are generated. Furthermore, it can be seen that configurations $(2|I|, 0.75)$ and $(2|I|, 0.85)$ -labels 2 and 3 respectively- tend to be non-dominated in most of the sampled instances. On the contrary, configurations $(3|I|, 0.85)$ and $(3|I|, 0.95)$ -labels 7 and 8- required large CPU times, but they did not find the best solutions in terms of quality. Note that for the 225×25 instances, all parameter configurations obtained the optimum value, hence CPU time was used to decide dominance among them.

Table 5.4: Dominance between the configurations.

	$(2\|I\|,0.65)$	$(2\|I\|,0.75)$	$(2\|I\|,0.85)$	$(2\|I\|,0.95)$	$(3\|I\|,0.65)$	$(3\|I\|,0.75)$	$(3\|I\|,0.85)$	$(3\|I\|,0.95)$
225×25-01	7	5	6	4	3	2	1	0
225×25-08	7	5	6	4	3	2	1	0
225×25-10	5	6	7	4	2	3	1	0
450×50-13	4	6	1	3	2	3	0	0
450×50-16	1	1	6	3	2	0	0	0
450×50-18	0	4	1	2	1	0	0	0
675×75-23	1	0	6	4	1	0	0	0
675×75-25	2	5	2	4	0	1	0	0
675×75-29	1	4	5	4	0	0	0	0
Total	28	36	40	32	14	11	3	0

Table 5.4 shows a summary of this analysis. It can be seen that configuration $(2|I|, 0.85)$ dominates most of other parameter configurations in the sampled instances. Based on this, we set the number of generations to two times the number of facilities in each instance, i.e. $2|I|$. The probability of selecting crossover operator was set to 0.85, encouraging the crossover

Figure 5.2: Dominance among configurations for sampled instances.



operator.

5.1.5 Computational experimentation

To evaluate the quality of the solutions obtained with the proposed GA, the reformulations described in section 5.1.2 were implemented using the commercial software CPLEX 12.6.1. Preliminary tests indicated that bounds obtained with the LP relaxation of the first reformulation are rather loose; hence, a considerable enumerative effort is needed by CPLEX to find optimal solutions for the test instances. A comparison between both reformulations showed that the second one provides tighter lower bounds, therefore, it was used to find optimal or near optimal values for the tested instances. Each instance was executed with a CPU time limit of three hours. The genetic algorithm was coded in Visual Studio 2010 in C++ language. All tests were performed using a personal computer having an Intel (R) Core processor with 8GB of RAM.

The values used for the algorithm's parameters were the ones detailed in the previous section, that is, P_{size}=100, $2|I|$ as the number of generations and 0.85 for the crossover probability.

Due to stochasticity immersed in the heuristic, each instance was executed ten times. The results obtained by the proposed GA are presented in Tables 5.5 and 5.6. The information in Tables 5.5 and 5.6 is the following, the first column is the instance name, the second column is the CPU time used by CPLEX to solve the instance (in all cases where this value is equal to 10,800, means that the optimal solution was not obtained in the time limit set for the tests). The next four columns show results obtained with the GA (percentage gaps of the best, average and worst solution values obtained by GA with respect to the optimal or best solution obtained with CPLEX within a three-hour time limit), the last column shows the average CPU time used by the proposed GA.

Table 5.5: Results for the instances in block M

	CPLEX	*GA*			
Instance	*CPU Time*	*Best*	*Average*	*Worst*	*CPU Time*
225×25-01	1.09	0.00	0.00	0.00	2.34
225×25-02	1.19	0.00	0.00	0.00	2.36
225×25-03	1.42	0.00	0.00	0.00	2.41
225×25-04	1.05	0.00	0.00	0.00	2.43
225×25-05	1.30	0.00	0.00	0.00	2.26
225×25-06	1.42	0.00	0.00	0.00	2.43
225×25-07	1.70	0.00	0.46	1.15	2.49
225×25-08	1.28	0.00	0.00	0.00	2.32
225×25-09	1.31	0.00	0.00	0.00	2.31
225×25-10	1.75	0.00	0.00	0.00	1.91
450×50-11	10.92	0.00	0.24	1.19	9.06
450×50-12	16.64	0.00	0.37	0.93	8.94
450×50-13	11.33	0.05	0.15	0.56	8.60
450×50-14	8.45	0.00	0.24	0.83	8.74
450×50-15	14.52	0.00	0.26	0.98	8.08
450×50-16	13.75	0.00	0.00	0.00	4.12
450×50-17	9.41	0.00	0.27	0.87	4.40
450×50-18	14.77	0.00	0.43	0.82	4.33
450×50-19	7.52	0.00	0.26	1.30	4.63
450×50-20	5.98	0.00	0.00	0.00	4.29
675×75-21	44.98	0.31	0.87	1.54	10.37
675×75-22	296.02	0.47	0.74	1.35	10.27
675×75-23	25.81	0.00	0.05	0.13	10.47
675×75-24	33.97	0.37	0.65	0.93	10.62
675×75-25	37.44	0.00	0.30	0.66	11.21
675×75-26	15.72	0.00	0.40	1.46	11.73
675×75-27	43.64	0.17	0.31	0.45	12.04
675×75-28	28.42	0.54	0.71	0.97	11.77
675×75-29	83.52	0.00	0.31	0.92	12.70
675×75-30	37.39	0.65	1.01	1.40	12.14

It can be observed in Table 5.5, that in the first subset of instances the GA always obtained the optimal value. Moreover, it can be seen that (with the exception of instance 225×25-07) the algorithm obtained the optimal solution in the ten runs. For the subset of instances with size 450×50, it can be noted that the algorithm was able to found the optimal value in 9 of the 10 instances. In the case when optimality was not reached (450×50-13), the percentage gap of the best solution obtained by the GA is 0.05%. For this second set of instances, the percentage gap never exceeded 1.3%. For the third subset of instances, the algorithm found optimal solutions for four out of the ten instances. In this case, the worst percentage gap never exceeded 1.54%. For this set of instances, the CPU time required by the genetic algorithm was smaller than the time required by CPLEX in all cases.

Table 5.6: Results for the instances in block L

	CPLEX	*GA*			
Instance	***CPU Time***	***Best***	***Average***	***Worst***	***CPU Time***
900×100-31	241.75	0.00	0.21	0.41	26.96
900×100-32	466.06	0.00	0.35	0.46	29.70
900×100-33	204.33	0.00	0.23	0.43	28.29
900×100-34	103.66	0.01	0.47	0.82	26.98
900×100-35	1342.30	0.02	0.24	0.39	28.85
900×100-36	176.20	0.10	0.37	0.60	27.77
900×100-37	3270.64	0.00	0.60	0.83	26.89
900×100-38	88.14	0.00	0.20	0.32	28.50
900×100-39	351.01	0.02	0.22	0.30	28.07
900×100-40	182.02	0.10	0.25	0.39	26.43
1350×150-41	10800.00	0.10	0.39	0.66	85.48
1350×150-42	10800.00	0.68	0.89	1.23	93.07
1350×150-43	10800.00	0.36	0.55	0.65	91.63
1350×150-44	10800.00	-0.45	-0.22	0.02	84.02
1350×150-45	10800.00	0.45	0.73	0.94	80.41
1350×150-46	309.64	0.35	1.51	2.02	83.38
1350×150-47	10800.00	0.05	0.60	0.87	83.63
1350×150-48	10800.00	2.29	2.48	2.60	82.46
1350×150-49	3952.89	0.41	0.63	0.81	84.70
1350×150-50	10800.00	0.76	1.38	1.77	89.86
1800×200-51	10800.00	0.90	1.18	1.48	231.45
1800×200-52	10800.00	0.64	0.91	1.05	251.10
1800×200-53	10800.00	-0.16	0.44	0.62	204.46
1800×200-54	10800.00	0.36	0.70	0.97	221.57
1800×200-55	10800.00	-0.30	-0.14	0.11	223.89
1800×200-56	10800.00	0.57	1.07	1.53	223.89
1800×200-57	10800.00	0.11	0.34	0.60	210.52
1800×200-58	10800.00	-0.50	-0.06	0.21	216.30
1800×200-59	10800.00	-0.03	0.27	0.58	228.00
1800×200-60	10800.00	0.76	1.16	1.44	236.82

With respect to larger instances, it can be clearly observed from Table 5.6 that the proposed GA required much less CPU time than CPLEX. In 18 instances CPLEX did not found the optimal value within three-hour time limit. In ten out of the 30 instances the proposed GA obtained the same or better lower bound than CPLEX. In general, we observed that the proposed GA is robust, since the worst average gap never exceeded 1.77%. We can also observe that for the set of 900×100 instances, the GA solved to optimality half of the instances and for the other half the worst optimality gap was 0.83%. For the second and third subsets, since the optimum is not known for most of the instances, the comparisons were made against the best known solution (optimal solution is known only for instances 1350×15-46 and 135×150-49). In those cases, when a negative value appears, it indicates that the GA outperformed CPLEX (five out of 18 instances).

5.1.6 A direct single-level reformulation

Also, a direct single-level reformulation that ensures the optimality of the follower's problem, which forces customer demand to be allocated to the most preferred facility is proposed. After performing preliminary computational tests with all the reformulations proposed in this chapter, it was observed that the linear relaxation bounds are much better for the direct single-level reformulation. The latter will be named as the single-level maximal covering location problem $(SLMCLP)$ reformulation.

Since the follower's problem ensures that the demand of customers that are within the coverage radius of one or more facilities is allocated to the most preferred facility, the bilevel mathematical program can be formulated as a single-level program. As in [26], we consider a single-level reformulation of the problem by replacing the follower's decision problem with a set of valid inequalities that ensure that whenever a customer is within the coverage radius of more than one facility, customer demand will always be allocated to the most preferred facility.

For each $j \in J$, let $i_1, i_2, \ldots, i_{|I(j)|}$ be the elements in $I(j)$ such that $g_{i_1,j} > g_{i_2,j} > \ldots > g_{i_{|I(j)|},j}$. Then, to ensure that the demand of each customer is allocated to the most preferred facility, the following valid inequalities are included:

$$\sum_{s=k+1}^{|I(j)|} x_{i_s,j} + y_{i_k} \leq 1 \qquad \forall j \in J, k \in 1, \ldots, |I(j)| - 1 \tag{5.77}$$

These inequalities ensure that if there is no open facility in the k^{th} preferred location for customer j, then the customer must be allocated to a facility whose location is less preferred than the k^{th} location or their demand is not allocated to any open facility. Then, the problem can be modelled with the following single-level mathematical programming model.

$$(SLMCLP) \quad \max_{y,x} \quad \sum_{i \in I_1} \sum_{j \in J(i)} D_j x_{ij} \tag{5.78}$$

$$\text{subject to:} \quad y_i = 1 \qquad \forall i \in I_2 \tag{5.79}$$

$$\sum_{i \in I_1} y_i = p \tag{5.80}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{5.81}$$

$$\sum_{i \in I(j)} x_{ij} = 1 \qquad \forall j \in J_2 \tag{5.82}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, j \in J(i) \tag{5.83}$$

$$\sum_{i \in I(j)} x_{ij} \leq 1 \qquad \forall j \in J_1 \tag{5.84}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J(i) \tag{5.85}$$

$$\sum_{s=k+1}^{|I(j)|} x_{i_s,j} + y_{i_k} \leq 1 \quad \forall j \in J, k \in 1, \ldots, |I(j)| - 1 \tag{5.86}$$

In the above formulation, we observe that constraints $x_{ij} \in \{0,1\}, i \in I, j \in J(i)$ can be replaced by $x_{ij} \geq 0, i \in I, j \in J(i)$. Given any feasible solution, where $I^* = \{i \in I : y_i = 1\}$ denotes the set of open facilities, for a given customer $j \in J$, let $i_1, i_2, \ldots, i_{|I(j)|}$ such that $g_{i_1 j} > g_{i_2 j} > \ldots > g_{i_{|I(j)|} j}$. Then, if $x_{i_k j} > 0$ by constraints (7), this implies that $y_{i_k} = 1$. In addition, by constraints (10) and (7), $y_{i_r} = 0$ and $x_{i_r j} = 0$ for all $r = 1, \ldots, k-1$. Otherwise, $x_{i_k j}$ cannot be greater than zero because if $y_{i_r} = 1$, for some $r = 1, \ldots, k-1, x_{i_k j}$ should be zero to satisfy constraints (10) associated with facility location i_r and customer j. Finally, $x_{i_r j} = 0$ for all $r = k+1, \ldots, |I(j)|$ given that $y_{i_k} = 1$. Therefore, by constraints (6) or

constraints (8) and assuming that $D_j > 0, x_{i_k j} = 1$.

Here, we prove the equivalence between $MCLP$ and $SLMCLP$.

Proposition 1 *If $(\overline{x}, \overline{y})$ is a feasible solution for $MCLP$, then valid inequalities (5.86) are satisfied and therefore $(\overline{x}, \overline{y})$ is also feasible for $SLMCLP$.*

Proof Let $\overline{x}$ be the optimal solution for the lower level problem for a given feasible solution of the leader's solution $\overline{y}$.

Therefore, in any feasible solution of the bilevel problem, for each customer $j \in J$ and a given $k \in [1, |I(j)|]$, we have the following cases.

1. $\overline{y}_{i_k} = 0$. We observe that in any feasible solution of $MCLP$, $\forall j \in J$, $\sum\limits_{i \in I(j)} x_{ij}$ is at most 1 since if $j \in J_2$, then $\sum\limits_{i \in I(j)} \overline{x}_{ij} = 1$ and if $j \in J_1$, then $\sum\limits_{i \in I(j)} \overline{x}_{ij} \leq 1$. Therefore, $\sum\limits_{s=k+1}^{|I(j)|} \overline{x}_{i_s,j} \leq 1$ since the set $\{i_{k+1}, \ldots, i_{|I(j)|}\} \subset I(j)$.

2. $\overline{y}_{i_k} = 1$. Therefore, $\overline{x}_{i_s,j} = 0$ for all $s > k$ since constraints (5.82)–(5.84) ensure that the demand of each customer is allocated to at most one open facility and if $\overline{x}_{i_s,j} = 1$ for some $s > k$, this will contradict the optimality of the lower level problem. In addition, since it is an uncapacitated problem and $g_{i_k,j} > g_{i_s,j}$ for all $s > k$, the demand of customer j can be allocated to their k^{th} preferred location because there is a facility located at that site (i.e., $\overline{y}_{i_k} = 1$).

Therefore, in both cases, constraints (5.86) hold. ∎

Proposition 2 *Let $(x^\star, y^\star)$ be an optimal solution of $SLMCLP$. Then, $x^\star$ is the optimal solution of the lower level problem in $MCLP$ for the given $y^\star$; therefore, $(x^\star, y^\star)$ is feasible for $MCLP$.*

Proof Let $I^\star = \{i \in I : y_i^\star = 1\}$. For each $j \in J$, we have the following mutually exclusive cases.

1. If $j \in J_2$, then the demand of customer j must be allocated to exactly one open facility because their demand is already covered by existing facilities. Therefore, let i_k be the facility to which the demand of customer j is allocated, i.e., $x^\star_{i_k,j} = 1$ for some $k \in [1, |I(j)|]$. Then, $x^\star_{i_s,j} = 0$ for all $s \in I(j) \setminus \{k\}$ because constraints (5.82) must hold. In addition, $y^\star_{i_k} = 1$ because $x^\star_{i_k,j} \leq y^\star_{i_k}$. Constraints (5.86) ensure that $y^\star_{i_s} = 0$ for all $s < k$ since if $y^\star_{i_s} = 1$ for some $s < k$, $x^\star_{i_k,j}$ must be equal to 0. Therefore, the demand of customer j is allocated to their most preferred facility among the open facilities, i.e., $i_k = \arg\max_{i \in I(j) \cap I^\star} \{g_{ij}\}$.

2. If $j \in J_1$, then the demand of customer j must be allocated to at most one open facility because constraints (5.84) must hold. Therefore, in the same way as in the case where $j \in J_1$, if the customer demand is allocated to facility i_k, this means that $x^{\star}_{i_k,j} = 1$ for some $k \in [1, |I(j)|]$ and $x^{\star}_{i_s,j} = 0$ for all $s \in I(j) \setminus \{k\}$. In addition, $\overline{y}_{i_k} = 1$ since $x^{\star}_{i_k,j} \leq y^{\star}_{i_k}$; thus, constraints (5.86) ensure that $y_{i_s} = 0$ for all $s < k$ and $i_k = \arg\max_{i \in I(j) \cap I^{\star}} \{g_{ij}\}$. If the demand of customer j is not allocated, this means that $y^{\star}_i = 0$ and $x^{\star}_{ij} = 0, \forall i \in I(j)$ because if $y^{\star}_k = 1$ for at least one $k \in I(j)$, this will contradict the optimality of the $SLMCLP$ because demands are assumed to be positive and therefore the objective value can be increased by D_j by setting $x^{\star}_{kj} = 1$. ∎

Proposition 3 *$MCLP$ and $SLMCLP$ provide the same optimal solutions.*

Proof Let $(x^{\star}, y^{\star})$ be an optimal solution for $MCLP$. By Proposition 1, we know that this solution is also feasible for $SLMCLP$. Assume that there exists an optimal solution $(\overline{x}, \overline{y})$ for the single-level formulation such that

$$\sum_{i \in I_1} \sum_{j \in J(i)} D_j \overline{x}_{ij} > \sum_{i \in I_1} \sum_{j \in J(i)} D_j x^{\star}_{ij}.$$

By Proposition 2, we know that this solution is also feasible for the bilevel formulation; therefore, it contradicts the optimality of $(x^{\star}, y^{\star})$ for $MCLP$.

Conversely, let $(x^{\star}, y^{\star})$ be an optimal solution for $SLMCLP$. By Proposition 2, we know that this solution is also feasible for $MCLP$. Assume that there is another feasible solution $(\overline{x}, \overline{y})$ for $MCLP$ such that

$$\sum_{i \in I_1} \sum_{j \in J(i)} D_j \overline{x}_{ij} > \sum_{i \in I_1} \sum_{j \in J(i)} D_j x^{\star}_{ij}.$$

By Proposition 1, $(\overline{x}, \overline{y})$ is also feasible for the $SLMCLP$, thereby contradicting the optimality of $(x^{\star}, y^{\star})$ for the $SLMCLP$. ∎

From the latter results, it can be concluded that both mathematical formulations are equivalent.

Note that this single-level reformulation can be used to obtain a reference value in order to measure the performance of the proposed algorithms.

5.1.7 GRASP and Hybrid GRASP-Tabu heuristics

Artificial intelligence (AI) has been applied in many domains, such as robotics, speech recognition, planning and programming for many tasks, logistics planning, pattern recognition, and VLSI design ([120]). When attempting to provide solutions to such problems, AI encounters many obstacles. One obstacle is related to the fact that many of the problems involve combinatorial problems in which finding all possible alternative solutions and investigating them is, in practical terms, impossible. In such situations, it is very useful to have informed search strategies that enable more efficient search processes. Heuristic and metaheuristic methods

help manage these complex decision problems by taking advantage of the structure and characteristics of the problems to be solved, thereby providing AI with efficient search methods ([60]).

The GRASP metaheuristic was first proposed by [50]. It is an iterative search procedure. In each iteration, a new solution is constructed by a greedy randomized procedure, which is then improved using a local search procedure. In [50], the authors give an intuitive justification for GRASP as a repetitive sampling technique. In each iteration of the greedy algorithm, a new element is selected from a *restricted candidate list* and added to the solution. The mean and variance of the sample distribution are a function of the cardinality of the restricted candidate list used in the constructive phase of GRASP. Since sample solutions are selected randomly by order statistics, one can intuitively expect that the best value found should outperform the mean value.

Tabu search ([60] and [61]) relies on the use of adaptive memory and responsive exploration. It uses adaptive memory to record historical information of the search process. The term responsive exploration refers to the ability of the method to make strategic choices to achieve effectiveness.

Memory structures used by tabu search operate by referencing four dimensions: recency, frequency, quality, and influence. The simplest forms of tabu search only use recency-based memory to avoid cycling. The memory structures are used to record the attributes of solutions recently visited, which are labeled tabu-active, and solutions that contain tabu-active attributes are forbidden. Therefore, the use of recency-based memory is an aggressive exploration strategy that attempts to go beyond local optimality because it prevents revisiting solutions visited in the recent past. Aspiration criteria can also be used to override tabu restrictions (remove tabu-active attributes). The simplest form of an aspiration criterion is to remove the tabu classification of a solution that is better than the best solution found so far.

Long-term memory is often used by a tabu search procedure for intensification or diversification purposes. Intensification strategies can be used to exploit features historically found to be good, while diversification strategies encourage the search process to explore unvisited regions of the solution space. Commonly, frequency-based memory is used to implement intensification and diversification strategies.

The quality dimension refers to the ability to differentiate the merit of solutions visited during the search, whereas the influence dimension refers to the impact of the choices made during the search process relative to both quality and structure of the visited solutions.

In the following, we describe the two proposed heuristics to find lower bounds for $MCLP$: a GRASP heuristic and a hybrid GRASP-Tabu heuristic, which is a combination of GRASP and tabu search.

5.1.7.1 GRASP Heuristic

The proposed GRASP heuristic iteratively constructs an initial feasible solution that is subsequently improved by a local search that explores one neighborhood. The incumbent solution is updated each time an iteration obtains a better solution. The algorithm terminates when the termination criterion is met. Here, we first describe the greedy randomized procedure used in the proposed GRASP heuristic and then describe the local search procedure.

Greedy Randomized Procedure

To find initial feasible solutions to $MCLP$, a set $S \subset I_1$ of p facilities must be selected such that customer demand covered by the firm is maximized. The greedy randomized procedure of the proposed GRASP heuristic is an iterative procedure that constructs an initial feasible solution. In each iteration, a facility from the set of potential facilities I_1 is selected and covered customers are assigned to their most preferred facility among the existing facilities. A greedy function is used to evaluate the contribution to the leader's objective function of each candidate facility in I_1. For each candidate facility, the greedy function measures the additional demand that can be covered by the facility by considering the follower's reaction. The demand that can be captured by a candidate facility is the demand of customers within its coverage radius and that is not covered by any open facility or a demand that is covered by some facility in I_2 but the customer prefers the candidate facility among all open facilities. Next, we explain the constructive procedure in detail.

Let $Covered$ be the set of customers covered by open facilities. Initially, $Covered$ is equal to J_2. In addition, $A : J \rightarrow S \cup I_2$, where $A(j) := i$ if customer j is allocated to facility i. For each $j \in J$, $A(j) := \arg\max_{i \in (S \cup I_2) \cap I(j)} \{g_{ij}\}$, if $(S \cup I_2) \cap I(j) \neq \emptyset$. Otherwise, $A(j) := null$, which means that customer j in not within the coverage radius of any open facility and therefore $j \notin Covered$. Initially, S is equal to $\emptyset$ and it is updated at the end of each iteration. To select the facility to be open at each iteration, the greedy value w_i is computed for each potential facility $i \in I_1 \setminus S$. As mentioned previously, this greedy value w_i measures the increase in the leader's objective function value to open a facility at location i. In other words, for each customer j covered by i (i.e., $j \in J(i)$), the demand is added to w_i in either of the following two situations: 1) j was not previously covered $j \notin Covered$, and 2) j was covered by a facility $A(j) \in I_2$ and i is more preferred than $A(j)$. For each $i \in I_1 \setminus S$, we have the following.

$$\beta_{ij} = \begin{cases} D_j & \text{if } j \notin Covered \text{ or } (g_{ij} > g_{A(j),j} \text{ and } A(j) \in I_2) \\ 0 & \text{otherwise} \end{cases} \quad ,\forall j \in J(i)$$

therefore

$$w_i = \sum_{j \in J(i)} \beta_{ij} \tag{5.87}$$

Let $w_{\min} = \min_{i \in I_1 \setminus S} \{w_i\}$ and $w_{\max} = \max_{i \in I_1 \setminus S} \{w_i\}$. In addition, let $T = w_{\min} + \alpha(w_{\max} - w_{\min})$ be a threshold value, where $\alpha \in (0, 1)$ is a parameter that controls the degree of greediness or randomness of the greedy procedure. We randomly select a facility i^* from a restricted candidate list RCL that contains candidate facilities whose $w_i \geq T$. Facility $i^\star$ is added to the set of open facilities S. At the end of each iteration, the $Covered$ set is updated and customers $j \in J(i^*)$ are allocated to i^* if they were not previously covered or if i^* is more preferred than its previous assignment $A(j)$. In addition, $A(j)$ is updated for all $j \in Covered$. In other words, $Covered$ is set to $\bigcup_{i \in (S \cup I_2)} J(i)$, which is the set of all customers within the coverage radius of some open facility. For all $j \in Covered$, $A(j)$ is set to $\arg\max_{i \in (S \cup I_2) \cap I(j)} \{g_{ij}\}$, which means that each covered customer is allocated to their most preferred location among all locations in $S \cup I_2$. The constructive phase ends when p facilities are selected. The randomized greedy procedure of the proposed GRASP heuristic is given in Algorithm 5.1.1.

Algorithm 5.1.1 Greedy randomized procedure

function GREEDYRANDOMIZED(α)
  $SolValue \leftarrow 0$
  $selected \leftarrow 0$
  $S \leftarrow \emptyset$
  $Covered \leftarrow J_2$
  for all $j \in J_2$ **do**
    $A(j) \leftarrow \arg\max_{i \in I_2} \{g_{ij} : j \in J(i)\}$
  end for
  repeat
    for all $i \in I_1 \setminus S$ **do**
      for all $j \in J(i)$ **do**

$$\beta_{ij} \leftarrow \begin{cases} D_j & \text{if } j \notin Covered \text{ or } (g_{ij} > g_{A(j),j} \text{ and } A(j) \in I_2) \\ 0 & \text{otherwise} \end{cases}$$

      end for
      $w_i \leftarrow 0$
      for all $j \in J_1$ **do**
        $w_i \leftarrow w_i + \beta_{ij}$
      end for
    end for
    $w_{\min} \leftarrow \min_{i \in I_1 \setminus S} \{w_i\}$
    $w_{\max} \leftarrow \max_{i \in I_1 \setminus S} \{w_i\}$
    $T \leftarrow w_{\min} + \alpha(w_{\max} - w_{\min})$
    $RCL \leftarrow \{i \in I_1 \setminus S : w_i \geq T\}$
    select i^* randomly from RCL
    $S \leftarrow S \cup \{i^*\}$
    $selected \leftarrow selected + 1$
    $Covered \leftarrow \bigcup_{i \in (S \cup I_2)} J(i)$
    $A(j) \leftarrow \arg\max_{i \in (S \cup I_2) \cap I(j)} \{g_{ij}\}$
    $DemandCovered \leftarrow DemandCovered + w_{i^*}$
  until $selected = p$
  return $SolValue$
end function

Local search Procedure

Each initial solution generated by the randomized greedy procedure of the GRASP heuristic is improved by a local search procedure. In this manner, the local search procedure in a GRASP heuristic is used as an intensification strategy [114]. In our case, the local search procedure explores the neighborhood wherein one open facility is exchanged with a closed facility, i.e., the exchange opens a closed facility $i_1 \in I_1 \setminus S$ and closes an open facility $i_2 \in S$. A solution to the maximal covering location problem with customer preference ordering is represented by a pair (S, A), where S is the set of open facilities and $A : J \rightarrow S \cup I_2$, as defined in the previous section. Therefore, the neighborhood $N(S, A)$ of a given solution (S, A)

is expressed as follows.

$$\begin{aligned} N(S,A) \;=\; & \{(S',A') : S' = S \setminus \{i_2\} \cup \{i_1\}, i_2 \in S, i_1 \in I_1 \setminus S, \\ & A'(j) = \arg\max_{i \in (S' \cup I_2) \cap I(j)} \{g_{ij}\}, \forall j \in \cup_{i \in (S' \cup I_2)} J(i), \\ & A'(j) = null, \forall j \notin \cup_{i \in (S' \cup I_2)} J(i)\} \end{aligned}$$

Here, $N(S,A)$ is the set of all solutions that can be obtained by opening a closed facility and closing an open facility.

Let

$$\delta_j^{i_1,i_2} = \begin{cases} -D_j, & \text{if } A(j) = i_2 \text{ and } (j \notin \cup_{i \in S \setminus \{i_2\} \cup \{i_1\}} J(i) \text{ or } \arg\max_{i \in (S \cup I_2) \setminus \{i_2\} \cup \{i_1\}} \{g_{ij}\} \in I_2) \\ D_j, & \text{if } j \in J(i_1) \text{ and } (j \notin \cup_{i \in (S \cup I_2)} J(i)) \text{ or } (A(j) \in I_2 \text{ and } g_{A(j),j} < g_{i_1,j})) \\ 0, & \text{otherwise} \end{cases}$$

for all $j \in J$, denote the leaders' objective function change associated with customer j when we exchange facility i_1 with facility i_2. As can be seen, the objective function value decreases by D_j if the demand of customer j is covered by facility i_2 in solution (S,A), i.e., $A(j) = i_2$ and

- the demand of customer j cannot be covered by the set of the leader's opened facilities $S \setminus \{i_2\} \cup \{i_1\}$ after the exchange is performed, or
- the demand of customer j will be allocated to some facility in I_2 according to preferences (i.e., $\arg\max_{i \in (S \cup I_2) \setminus \{i_2\} \cup \{i_1\}} \{g_{ij}\} \in I_2$).

On the other hand, the objective function value will increase by D_j when the demand of customer j is covered by facility i_1 (i.e., $j \in J(i_1)$) and

- the demand of customer j is not covered in the current solution (i.e., $j \notin \cup_{i \in (S \cup I_2)} J(i)$), or
- the demand of customer j is allocated to some facility in I_2; however, according to customer preferences, it will be reallocated to facility i_1 because $g_{A(j),j} < g_{i_1,j}$.

Then, the leader's objective function change associated with the exchange of facility i_1 with facility i_2 is expressed as follows.

$$\Delta_{i_1,i_2} = \sum_{j \in J} \delta_j^{i_i,i_2}$$

Here, Δ_{i_1,i_2} measures the impact to the leader's objective function value by computing the customer's demand change when facility $i_1 \in I_1 \setminus S$ is opened and the impact to the customer's demand change when facility $i_2 \in S$ is closed by considering the follower's reaction with respect to customer preferences.

The local search of the proposed GRASP heuristic uses the best improvement strategy (i.e., the best solution in $N(S,A)$ is selected) and terminates when the objective function value cannot be improved further (all solutions in $N(S,A)$ are worse than the current solution). The local search procedure of

the proposed GRASP heuristic is given in Algorithm 5.1.2.

Algorithm 5.1.2 Local search procedure

function LOCALSEARCH($SolValue$)
  $Best \leftarrow SolValue$
  $Stop \leftarrow$ **false**
  repeat
    for all $i_2 \in S$ and $i_1 \in I_1 \setminus S$ **do**
      for all $j \in J$ **do**

$$\delta_j^{i_1,i_2} = \begin{cases} -D_j, & \text{if } A(j) = i_2 \text{ and } (j \notin \cup_{i \in S \setminus \{i_2\} \cup \{i_1\}} J(i) \text{ or } \arg\max_{i \in (S \cup I_2) \setminus \{i_2\} \cup \{i_1\}} \{g_{ij}\} \in I_2) \\ D_j, & \text{if } j \in J(i_1) \text{ and } (j \notin \cup_{i \in (S \cup I_2)} J(i)) \text{ or } (A(j) \in I_2 \text{ and } g_{A(j),j} < g_{i_1,j})) \\ 0, & \text{otherwise} \end{cases}$$

      end for
      $\Delta_{i_1,i_2} \leftarrow 0$
      for all $j \in J$ **do**
        $\Delta_{i_1,i_2} \leftarrow \Delta_{i_1,i_2} + \delta_j^{i_i,i_2}$
      end for
    end for
    $\Delta \leftarrow \max\{\Delta_{i_1,i_2} : i_2 \in S \text{ and } i_1 \in I_1 \setminus S\}$
    if $\Delta > 0$ **then**
      $(i_1^*, i_2^*) \in \arg\max_{i_2 \in S, i_1 \in I_1 \setminus S}\{\Delta_{i_1,i_2}\}$
      $S \leftarrow S \cup \{i_1^*\} \setminus \{i_2^*\}$
      $Best \leftarrow Best + \Delta$
      for all $j \in J$ **do**
        $A(j) \leftarrow \arg\max_{i \in (S \cup I_2) \cap I(j)} \{g_{ij}\}$
      end for
    else
      $Stop \leftarrow$ **true**
    end if
  until $Stop$
  return $Best$
end function

The proposed GRASP heuristic is given in Algorithm 5.1.3. The termination criterion is a predetermined number of iterations without improvement.

Algorithm 5.1.3 GRASP heuristic

function GRASP($NumberOfIterations, \alpha_{initial}, \alpha_{final}, ItersFixed, Increment$)
  $IterCount \leftarrow 1$
  $\alpha \leftarrow \alpha_{initial}$
  $Best \leftarrow 0$
  while ($IterCount \leq NumberOfIterations$) **do**
    if ($IterCount$ **mod** $Iterfixed = 0$) **then**
      if ($\alpha = \alpha_{final}$) **then**
        $\alpha \leftarrow \alpha_{initial}$
      else
        $\alpha \leftarrow \alpha + Increment$
      end if
    end if
    $SolValue \leftarrow GreedyRandomized(\alpha)$
    $SolValue \leftarrow LocalSearch(SolValue)$
    if ($SolValue > Best$) **then**
      $Best \leftarrow SolValue$
      $IterCount \leftarrow 0$
    end if
    $IterCount \leftarrow IterCount + 1$
  end while
  return $Best$
end function

5.1.7.2 Hybrid GRASP-Tabu Heuristic

Hybrid algorithms have been widely used to solve difficult problems ([17], [88], and [130]). These combinations enhance the advantages of using a single methodology to obtain better solutions. The combination of GRASP with Tabu search was first studied in [80]. Hybrid methods using GRASP with Tabu search have been used to find feasible bounds for many problems ([1], [30], [36], [41], [44], [80], and [85]). In this study, we propose a procedure that combines GRASP and Tabu search. The Tabu search procedure is used instead of the local search in the algorithm's improvement phase.

Tabu search is commonly used to solve difficult problems. It was first proposed in [59] and it is described in detail in [60]. In this methodology, short-term and long-term memory are used to escape local optima in order to better explore the solution space. Three elements must be defined to implement tabu search: tabu attributes, tabu-tenure (a parameter that indicates the number of iterations in which an attribute will be active), and the aspiration criterion (criterion to override tabu-active attributes). Tabu-active attributes are those that are forbidden in a solution. The short-term memory of a tabu search contains a list of selected attributes that occur in recently visited solutions to prevent revisiting these solutions.

Algorithm 5.1.4 Tabu search procedure

1: **function** TABUSEARCH($SolValue$)
2: $Best \leftarrow SolValue$
3: $StopCriterion \leftarrow$ **false**
4: **repeat**
5: **for all** $i_2 \in S$ and $i_1 \in I_1 \setminus S$ **do**
6: **for all** $j \in J$ **do**
7:
$$\delta_j^{i_1,i_2} = \begin{cases} -D_j, & \text{if } A(j) = i_2 \text{ and } (j \notin \cup_{i \in S \setminus \{i_2\} \cup \{i_1\}} J(i) \text{ or } \arg\max_{i \in (S \cup I_2) \setminus \{i_2\} \cup \{i_1\}} \{g_{ij}\} \in I_2) \\ D_j, & \text{if } j \in J(i_1) \text{ and } (j \notin \cup_{i \in (S \cup I_2)} J(i)) \text{ or } (A(j) \in I_2 \text{ and } g_{A(j),j} < g_{i_1,j})) \\ 0, & \text{otherwise} \end{cases}$$
8: **end for**
9: $\Delta_{i_1,i_2} \leftarrow 0$
10: **for all** $j \in J$ **do**
11: $\Delta_{i_1,i_2} \leftarrow \Delta_{i_1,i_2} + \delta_j^{i_i,i_2}$
12: **end for**
13: **end for**
14:
$$Candidates \leftarrow \{(i_1, i_2) : i_1 \in I_1 \setminus S, i_2 \in S \text{ and } (i_2 \notin TabuList \text{ or } SolValue + \Delta_{i_1,i_2} > Best)\}$$
15: $\Delta \leftarrow \max\{\Delta_{i_1,i_2} : (i_1, i_2) \in Candidates\}$
16: $(i_1^*, i_2^*) \in \arg\max\{\Delta_{i_1,i_2} : (i_1, i_2 \in Candidates\}$
17: $S \leftarrow S \cup \{i_1^*\} \setminus \{i_2^*\}$
18: **for all** $j \in J$ **do**
19: $A(j) \leftarrow \arg\max_{i \in (S \cup I_2) \cap I(j)} \{g_{ij}\}$
20: **end for**
21: Update $TabuList$
22: $SolValue \leftarrow SolValue + \Delta$
23: **if** $SolValue > Best$ **then**
24: $Best \leftarrow SolValue$
25: **end if**
26: Update $StopCriterion$
27: **until** (**not** $StopCriterion$)
28: **return** $Best$
29: **end function**

In our implementation, the tabu attributes are defined in a simple manner. Once a facility $i \in S$ is closed, it must remain closed for a number of iterations. To implement this, a fixed size list with a first in-first out discipline is maintained. Therefore, if facility $i^\star \in S$ is closed at iteration t, the index of facility $i^\star$ will be appended to the tabu list. If the fixed size of the tabu list is r, opening facility $i^\star$ will be forbidden in iterations $t+1, t+2, \ldots, t+r$. We also use the standard aspiration criterion every time a solution with a tabu-active attribute is better than the incumbent solution, i.e., the tabu restrictions are ignored. Therefore, at any iteration of the procedure, the candidate neighbor solutions will be those that do not contain tabu-active attributes or satisfy the aspiration criterion (line 14 in Algorithm 5.1.4). The proposed hybrid GRASP-Tabu heuristic is given in Algorithm 5.1.5. The termination criterion is a predetermined number of iterations without improvement.

Algorithm 5.1.5 Hybrid GRASP-Tabu heuristic

function HYBRID($NumberOfIterations, \alpha_{initial}, \alpha_{final}, ItersFixed, Increment$)
  $IterCount \leftarrow 1$
  $\alpha \leftarrow \alpha_{initial}$
  $Best \leftarrow 0$
  while ($IterCount \leq NumberOfIterations$) **do**
    if ($IterCount$ **mod** $Iterfixed = 0$) **then**
      if ($\alpha = \alpha_{final}$) **then**
        $\alpha \leftarrow \alpha_{initial}$
      else
        $\alpha \leftarrow \alpha + Increment$
      end if
    end if
    $SolValue \leftarrow GreedyRandomized(\alpha)$
    $SolValue \leftarrow TabuSearch(SolValue)$
    if ($SolValue > Best$) **then**
      $Best \leftarrow SolValue$
      $IterCount \leftarrow 0$
    end if
    $IterCount \leftarrow IterCount + 1$
  end while
  return $Best$
end function

5.1.8 Additional computational experimentation

To evaluate the performance of the proposed heuristic methods, we use the same set of instances described in section 5.1.4. To evaluate the quality of the solutions obtained by the proposed heuristics, the single-level reformulation was coded and run using the FICO XPRESS 7.8 commercial software. Both heuristics (the GRASP heuristic and hybrid GRASP-Tabu heuristic) were coded in C++. The single-level reformulation and both heuristics were run on an Intel Xenon(R) processor at 3.10 GHz with 8 GB of RAM. The computational experimentation for both heuristics and the obtained results are discussed below.

The FICO XPRESS Optimization Suite provides a mathematical programming framework with different algorithms to solve LP problems and mixed integer LP problems (MIP). For LP problems, it includes the primal simplex, dual simplex, and Newton barrier algorithms. For MIP problems, the solver provides a powerful branch and bound framework that uses heuristic methods to quickly determine good solutions and cutting planes to strengthen the LP relaxations. The use of heuristics and cutting planes to provide primal and dual bounds might allow considerable reduction of enumerative effort. In addition, the MIP solver uses pre-solve procedures that might help reduce the problem matrix and in turn the dimension of the problem, thereby making it easier to solve. In preliminary tests with FICO XPRESS, it was observed that, for the largest instances, executing single-level reformulation for each instance could take more than 15 hours. In addition, in preliminary tests, it was observed that executing the proposed heuristics for each instance never exceeded 250 seconds. Thus, a time limit of three hours (two orders of magnitude larger than the time required by the heuristics for the largest

instances) was set to run the single-level reformulation in FICO XPRESS.

Here, an evaluation of the proposed GRASP heuristic is described. The GRASP heuristic has two parameters that need to be adjusted: the α value and the termination criterion (number of iterations without improvement). The selection of the α parameter in the construction phase of a GRASP heuristic may affect the solution quality; therefore, it is very important to select an appropriate value for α. However, due to difficulty in setting a value for α that is suitable for all instances of a given problem, several options have been used in previous works. A trial-and-error strategy is often used to find the most suitable parameter value for each data instance. Other implementations ([106], [36], and [117]) automatically tune the α value using a reactive procedure. These procedures are usually based on probabilities to select α values from a discrete set of values, i.e., the α value used in each iteration. For these procedures to work correctly, the heuristic algorithm must perform many iterations. In this study, the solution quality of the proposed algorithms is sufficient without requiring many iterations; thus, a reactive procedure is not necessary. Instead, in all tests, the α parameter was varied from 0.75 to 0.95, starting at 0.75 and increasing α by 0.05 every five iterations.

The criterion used to terminate the GRASP heuristic is a fixed number of iterations without improvement. Two sets of tests were performed to evaluate the quality of the results obtained with the proposed GRASP heuristic and heuristic robustness. In the first set of tests, the termination criterion was 50 iterations without improvement, and in the second set of tests, the termination criterion was 100 iterations without improvement. Each set of tests consisted of running each test instance five times. To evaluate solution quality, the best solution from the five runs was compared to the optimum or best known solution of each instance, and, to evaluate heuristic robustness, the deviation between the best and worst solutions was measured.

Tables 5.7 and 5.8 describe the results obtained with the proposed GRASP heuristic. The first column shows the size of the instances, and the next three columns show the average percentage deviation of the best solutions obtained in each instance with respect to the optimal or best known solution (Avg. Best %), the average percentage deviation of the solution's mean with respect to the optimal or best known solution (Avg. Mean %), and the average percentage deviation of the worst solutions obtained with respect to the optimal or best known solution (Avg. Worst %). Finally, the last two columns show the average CPU time in seconds required by the GRASP heuristic and XPRESS, respectively.

Note that, for instances of size 1800×200, XPRESS was not able to find the optimal solution in the three-hour time limit for nine out of the ten instances. In addition, for these nine instances, the best feasible solution value obtained by XPRESS was worse than the solution obtained by the GRASP heuristic. In both sets of tests, the algorithm found the optimal or best known solution for all instances in at least one out of the five runs. Moreover, the average percentage deviation of the worst solutions obtained with respect to the optimal or best known solution never exceeded 0.032% for the experiment with termination criterion of 50 iterations without improvement and 0.031% for the experiment with 100 iterations. Clearly, the algorithm is more robust by increasing the number of iterations without improvement in the termination criterion given that the deviation between the best and the worst solutions is reduced.

Table 5.9 shows the results of the proposed hybrid GRASP-Tabu heuristic. As with the GRASP heuristic tests, the α parameter was varied from 0.75 to 0.95, starting at 0.75 and increasing by 0.05 every five iterations. In addition, for this test, the number of iterations without improvement was set to 50, and Tabu tenure was set to seven iterations. Again, the test consisted of five runs for each test instance.

Table 5.7: GRASP Heuristic Results (50 Iterations)

Size	Avg. Best %	Avg. Mean %	Avg. Worst %	CPU (sec)	XPRESS CPU(sec)
225 x 25	0.000	0.000	0.000	0.0	1.9
450 x 50	0.000	0.001	0.005	0.0	6.4
675 x 75	0.000	0.018	0.031	1.5	21.1
900 x 100	0.000	0.002	0.010	4.5	65.3
1350 x 150	0.000	0.004	0.018	24.1	2562.6
1800 x 200	0.000	0.011	0.032	79.1	>10800.0

Table 5.8: GRASP Heuristic Results (100 Iterations)

Size	Avg. Best %	Avg. Mean %	Avg. Worst %	CPU (sec)	XPRESS CPU (sec)
225 x 25	0.000	0.000	0.000	0.0	1.9
450 x 50	0.000	0.000	0.000	0.2	6.4
675 x 75	0.000	0.018	0.031	3.1	21.1
900 x 100	0.000	0.002	0.010	9.0	65.3
1350 x 150	0.000	0.001	0.003	44.3	2562.6
1800 x 200	0.000	0.005	0.015	142.6	>10800.0

To evaluate the solution quality, the best solution from the five runs was compared with the optimum or best known solution of each instance, and, to evaluate heuristic robustness, the deviation between the best and worst solutions was measured. Clearly, the hybrid GRASP-Tabu heuristic gives the best quality results. The average percentage deviation of the solution's mean with respect to the optimal or best known solution in the worst case was 0.006%, which is much lower compared with the 0.018% of the GRASP heuristic. In addition, the average percentage deviation of the worst solutions obtained with respect to the optimal or best known solution was improved substantially for instances greater than 675×75. In only three out of the 60 instances, the algorithm did not find the optimal or best known solution in the five runs, clearly showing that the algorithm is robust. Finally, although the average CPU time was greater for the hybrid GRASP-Tabu heuristic, the average time only increased slightly than the CPU time of the GRASP heuristic and remained much lower than the CPU time of the mathematical model run in FICO XPRESS.

The detailed results for each instance are shown in Table 5.10. The first column shows the name of each instance, and the second column shows the optimal or best known solution. The following three columns show the best, average, and worst percentage deviations of the solutions with respect to the optimal or best known solution. The last column shows the average CPU time in seconds.

Table 5.9: Hybrid GRASP-Tabu Results

Size	Avg. Best %	Avg. Mean %	Avg. Worst %	CPU (sec)	XPRESS CPU (sec)
225 x 25	0.000	0.000	0.000	0.0	1.9
450 x 50	0.000	0.000	0.000	1.4	6.4
675 x 75	0.000	0.006	0.031	7.6	21.1
900 x 100	0.000	0.000	0.000	13.9	65.3
1350 x 150	0.000	0.000	0.000	58.8	2562.6
1800 x 200	0.000	0.001	0.001	172.8	$>$10800.0

Table 5.10: Detailed Hybrid GRASP-Tabu Results

Name	Optimal or Best Known	Best %	Avg. %	Worst %	CPU
mcb225x25-01	20870	0.000	0.000	0.000	< 1.00
mcb225x25-02	18562	0.000	0.000	0.000	< 1.00
mcb225x25-03	19546	0.000	0.000	0.000	< 1.00
mcb225x25-04	21236	0.000	0.000	0.000	< 1.00
mcb225x25-05	20160	0.000	0.000	0.000	< 1.00
mcb225x25-06	19458	0.000	0.000	0.000	< 1.00
mcb225x25-07	20243	0.000	0.000	0.000	< 1.00
mcb225x25-08	20511	0.000	0.000	0.000	< 1.00
mcb225x25-09	19815	0.000	0.000	0.000	< 1.00
mcb225x25-10	20210	0.000	0.000	0.000	< 1.00
mcb450x50-11	73623	0.000	0.000	0.000	1.80
mcb450x50-12	79903	0.000	0.000	0.000	1.20
mcb450x50-13	76999	0.000	0.000	0.000	2.20
mcb450x50-14	68361	0.000	0.000	0.000	1.20
mcb450x50-15	73454	0.000	0.000	0.000	1.60
mcb450x50-16	72540	0.000	0.000	0.000	1.20
mcb450x50-17	79528	0.000	0.000	0.000	1.20
mcb450x50-18	75316	0.000	0.000	0.000	1.20
mcb450x50-19	74286	0.000	0.000	0.000	1.00
mcb450x50-20	80620	0.000	0.000	0.000	1.20
mcb675x75-21	182743	0.000	0.000	0.000	7.40
mcb675x75-22	176540	0.000	0.061	0.306	10.40
mcb675x75-23	183008	0.000	0.000	0.000	7.00
mcb675x75-24	177188	0.000	0.000	0.000	8.20
mcb675x75-25	185892	0.000	0.000	0.000	8.00
mcb675x75-26	176562	0.000	0.000	0.000	6.00
mcb675x75-27	201912	0.000	0.000	0.000	6.80
mcb675x75-28	188272	0.000	0.000	0.000	6.20
mcb675x75-29	203018	0.000	0.000	0.000	8.80
mcb675x75-30	183625	0.000	0.000	0.000	7.20
mcb900x100-31	300826	0.000	0.000	0.000	12.60
mcb900x100-32	325846	0.000	0.000	0.000	12.60
mcb900x100-33	328118	0.000	0.000	0.000	15.00
mcb900x100-34	309788	0.000	0.000	0.000	11.60
mcb900x100-35	318625	0.000	0.000	0.000	15.20
mcb900x100-36	319178	0.000	0.000	0.000	13.40
mcb900x100-37	314034	0.000	0.000	0.000	15.80
mcb900x100-38	327188	0.000	0.000	0.000	12.00
mcb900x100-39	317017	0.000	0.000	0.000	17.80
mcb900x100-40	298779	0.000	0.000	0.000	13.20
mcb1350x150-41	693053	0.000	0.000	0.000	59.20
mcb1350x150-42	723185	0.000	0.000	0.000	65.80
mcb1350x150-43	742178	0.000	0.000	0.000	61.20
mcb1350x150-44	688609	0.000	0.000	0.000	66.00
mcb1350x150-45	684277	0.000	0.000	0.000	61.80
mcb1350x150-46	695860	0.000	0.000	0.000	44.00
mcb1350x150-47	690537	0.000	0.000	0.000	48.60
mcb1350x150-48	693398	0.000	0.000	0.000	61.60
mcb1350x150-49	731687	0.000	0.000	0.000	50.60
mcb1350x150-50	723746	0.000	0.000	0.000	69.00
mcb1800x200-51	1281284	0.000	0.000	0.000	144.40
mcb1800x200-52	1356246	0.000	0.006	0.010	211.80
mcb1800x200-53	1258479	0.000	0.000	0.000	163.80
mcb1800x200-54	1314660	0.000	0.001	0.004	168.60
mcb1800x200-55	1300838	0.000	0.000	0.000	145.20
mcb1800x200-56	1256017	0.000	0.000	0.000	157.60
mcb1800x200-57	1290808	0.000	0.000	0.000	206.20
mcb1800x200-58	1280134	0.000	0.000	0.000	180.20
mcb1800x200-59	1287646	0.000	0.000	0.000	201.00
mcb1800x200-60	1331171	0.000	0.000	0.000	149.20

Chapter 6

Capacitated Facility Location Problems

In the case when customer preferences are taken into account, the uncapacitated facility location problem that minimizes the total cost has a natural bilevel formulation. Since the lower level corresponds to an allocation problem, the bilevel problem can be easily reformulated as a single-level one. This single-level formulation is not so evident when capacity constraints are considered. First, it is necessary to specify the meaning of the capacity constraint. Moreover, as in ([133]), different assignment constraints can be considered. So, it is interesting to explore these alternatives to identify under which assumptions the original bilevel problem can be formulated through an adequate set of constraints as a single-level problem. Otherwise, it would be necessary to study properties of the bilevel feasible set.

The capacity constraints can be given in terms of number of customers or amount of demand. These constraints can refer to a minimum of demand satisfied by a facility to be open or a maximum of demand attended by each facility. It is possible that the leader wishes to open a facility but none of the customers prefer it, so it does not have sense to open it.

In this chapter we presented two models. In the former, unitary demand is considered and the lower level problem is converted into the transportation problem. In the latter, generalized demand is considered. In this case, the the lower level problem is the well-known generalized assignment problem (GAP). In the first case, the lower level problem can be optimally solved, but in the second case the complexity of the lower level problem demands alternatives for obtaining -in general- the follower's rational reaction set. Hence, the bilevel attainable solutions are defined for solving this particular bilevel problem in an efficient manner.

The remainder of the chapter is structured as follows: Section 6.1.1 states the mathematical formulation of the bilevel capacitated facility location problem with unitary demand. In Section 6.1.2, an equivalent single-level reformulation is proposed. Section 6.1.3 describes the proposed algorithm, which is a genetic algorithm. Then, the computational experimentation and its corresponding interpretation of the results are shown in Section 6.1.4. After that, the case when the demand is generalized is analyzed; hence, the bilevel formulation of the corresponding problem is presented in Section 6.2.1. In Section 6.2.2, the definitions of the necessary concepts for approximating the inducible region are presented. Section 6.2.3 presents a bound for the bilevel problem based on a reformulation considering the linear relaxation of the lower level problem. An analysis of the difficulty in obtaining upper or lower bounds in bilevel programming (in particular in this problem) is made in Section 6.2.4. In Section 6.2.5, the description of the heuristic algorithm implemented for solving this version of the problem is detailed. Computational experimentation is given in Section 6.2.6. In particular, the methodology followed is

explained and a comparison among the different versions of the proposed algorithms' performance is presented.

6.1 Capacitated facility location problem with unitary demand

6.1.1 Mathematical model

Let I denotes the facilities and J the customers, where $I = 1, ..., n$ and $J = 1, ..., m$. Let c_{ij} represent the costs for supplying the demand of the customer j by the facility i. Also, f_i denotes the fixed cost for opening the i-th facility. Each customer j has a preference to be served by facility i, represented by g_{ij}. Finally, q_i represents the capacity in facility i in terms of the maximum number of customers that can be attended in a facility. The binary decision variables of the bilevel problem are: y_i that represent whether the facility i is opened or not and x_{ij} that represent whether facility i satisfies the demand of the customer j.

Then, the mathematical model would be as follows:

$$\min_{y} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} f_i y_i \tag{6.1}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} q_i y_i \geq m \tag{6.2}$$

$$y_i \in \{0,1\} \qquad i = 1, ..., n \tag{6.3}$$

where x_{ij} is the optimal solution of the problem:

$$\min_{x} \sum_{i=1}^{n}\sum_{j=1}^{m} g_{ij}x_{ij} \tag{6.4}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, ..., m \tag{6.5}$$

$$\sum_{j=1}^{m} x_{ij} \leq q_i y_i \qquad i = 1, ..., n \tag{6.6}$$

$$x_{ij} \in \{0,1\} \qquad i = 1, ..., n, j = 1, ..., m \tag{6.7}$$

In this case, despite the fact that the preferences are given as ordered consecutive integer numbers, the uniqueness in the follower's problem is not guaranteed. The latter is caused by the capacities considered in each facility. Hence, in order to have well posed the bilevel problem the optimistic version is considered. In other words, if multiple optimal solutions for a given leader's decision can be found, then the one which incurs in the best leader's objective function value is selected. The specific procedure for obtaining all the multiple optimal follower's solutions is explained in the section that corresponds to the algorithm's description.

6.1.2 Single-level reformulation

Due to the lack of a commercial software capable to solve the bilevel problem described in the previous section, other resolution schemes are explored. One of the most common approaches is to reformulate

the lower level problem. We developed a reformulation of the bilevel problem by reducing it to a single-level mixed-integer programming problem. As we mentioned earlier, under the assumptions herein considered, the lower level is a transportation problem with n facilities with an offer of 0 or q_i depending on the leader's variables and m customers with demand equals to 1. We study properties on the optimal solution of the following problem in order to characterize the inducible region:

$$P(y): \quad \min_x \sum_{i=1}^{n}\sum_{j=1}^{m} g_{ij}x_{ij} \tag{6.8}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, ..., m \tag{6.9}$$

$$\sum_{j=1}^{m} x_{ij} \leq q_i y_i \qquad i = 1, ..., n \tag{6.10}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, ..., n, j = 1, ..., m \tag{6.11}$$

Now, let us formulate the lower level problem in another way. Let $I = \{i \in \{1, ..., n\} : y_i = 1\}$. That is, I is the set of open facilities in a solution y of the upper level problem. Then, the problem can be formulated as follows.

$$\min_x \sum_{i=1}^{n}\sum_{j=1}^{m} g_{ij}x_{ij} \tag{6.12}$$

$$\text{subject to:} \quad \sum_{i\in I} x_{ij} \geq 1 \qquad j = 1, ..., m \tag{6.13}$$

$$-\sum_{j=1}^{m} x_{ij} \geq -q_i \qquad i \in I \tag{6.14}$$

$$x_{ij} \geq 0 \qquad i \in I, j = 1, ..., m \tag{6.15}$$

In can be noted that equation (6.13) now is an inequality. This change does not affect the model due to the sense of the objective function, which is of minimization. Therefore, the variable x_{ij} will take the minimum value, that is, 1. Equation (6.14) is multiplied by a -1. In addition, the variable y_i is omitted because in this problem only the open facilities are being considered. Also, the integrity constraint of the variables x_{ij} is changed by (6.15), then the lower level corresponds to a linear programming problem.

The dual of lower level problem is defined by:

$$\max_{u,v} \sum_{j=1}^{m} u_j - \sum_{i\in I} q_i v_i \tag{6.16}$$

$$\text{subject to:} \quad u_j - v_i \leq g_{ij} \qquad i \in I, j = 1, ..., m \tag{6.17}$$

$$u_j \geq 0 \qquad j = 1, ..., m \tag{6.18}$$

$$v_i \geq 0 \qquad i \in I \tag{6.19}$$

Note that, in the reformulation of the lower level problem, the variables $\{x_{ij}, i \notin I, j = 1, ..., m\}$ are not included since $x_{ij} \leq y_i$ and their value is automatically assigned once the leader variables are known. This fact implies that in the dual problem, only $card(I) + m$ dual variables are defined. By duality theory ([12]), if one problem possesses an optimal solution, then both problems possess

optimal solutions and both optimal objective values are equal. In this case, the lower level problem has a bounded optimal solution since it is a transportation problem with an offer greater or equal to the number of customers.

Let $\{x_{ij}, i \in I, j = 1, ..., m\}$ verifying (6.13)-(6.15) and $\{f(u_j, v_i), i \in I, j = 1, ..., m\}$ verifying (6.17)-(6.19) such that:

$$\sum_{i \in I} \sum_{j=1}^{m} g_{ij} x_{ij} = \sum_{j=1}^{m} u_j - \sum_{i \in I} q_i v_i \tag{6.20}$$

Then, the vector (x, u, v) satisfies the complementary slackness conditions and x and (u, v) are optimal solutions to the primal and dual problems, respectively. Therefore, for each value of y, decision variables controlled by the leader, (y, x) provides a bilevel feasible solution if and only if there exists (u, v) such that:

$$\begin{aligned}
x_{ij} &= 0, \qquad i \notin I, j = 1, ..., m && (6.21)\\
\sum_{i \in I} x_{ij} &\geq 1 \qquad j = 1, ..., m && (6.22)\\
\sum_{j=1}^{m} x_{ij} &\leq q_i \qquad i \in I && (6.23)\\
u_j - v_i &\leq g_{ij} \qquad i \in I, j = 1, ..., m && (6.24)\\
u_j &\geq 0 \qquad j = 1, ..., m && (6.25)\\
v_i &\geq 0 \qquad i \in I && (6.26)\\
\sum_{i \in I} \sum_{j=1}^{m} g_{ij} x_{ij} &= \sum_{j=1}^{m} u_j - \sum_{i \in I} q_i v_i && (6.27)
\end{aligned}$$

Bearing in mind (6.21)-(6.27), the problem under study can be reformulated as follows:

$$\min_{y,x,u,v} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} f_i y_i \tag{6.28}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} q_i y_i \geq m \tag{6.29}$$

$$x_{ij} \leq y_i \qquad i = 1,...,n, j = 1,...,m \tag{6.30}$$

$$\sum_{i=1}^{n} x_{ij} \geq 1 \qquad j = 1,...,m \tag{6.31}$$

$$\sum_{j=1}^{m} x_{ij} \leq q_i y_i \qquad i = 1,...,n \tag{6.32}$$

$$u_j - v_i \leq g_{ij} + M(1 - y_i) \qquad i = 1,...,n, j = 1,...,m \tag{6.33}$$

$$v_i \leq M y_i \qquad i = 1,...,n \tag{6.34}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} g_{ij}x_{ij} - \sum_{j=1}^{m} u_j + \sum_{i=1}^{n} q_i v_i = 0 \tag{6.35}$$

$$y_i \in \{0,1\}, \qquad i = 1,...,n \tag{6.36}$$

$$x_{ij} \in \{0,1\}, \qquad i = 1,...,n, j = 1,...,m \tag{6.37}$$

$$u_j \geq 0, \qquad j = 1,...,m \tag{6.38}$$

$$v_i \geq 0, \qquad i = 1,...,n \tag{6.39}$$

It is important to remark that this reformulation is equivalent to the original bilevel problem introduced in the previous section.

6.1.3 Proposed algorithm

Due to the existing complexity when solving bilevel programming problems, many heuristic and metaheuristic algorithms have been proposed for obtaining optimal or good quality solutions. One metaheuristic that has been extensively applied to this kind of problems is the genetic algorithm. For example, in [86] a genetic algorithm for solving Stackelberg-Nash equilibrium of nonlinear multilevel programming with multiple followers in which there might be information exchange among the followers is designed. That genetic algorithm might involve some iterative process or evolution subprocess for solving Nash equilibrium of followers for each control vector revealed by the leader. In [68] and [84] the lower level problem is replaced by the Karush-Kuhn-Tucker optimality conditions yielding a single-level optimization problem, a genetic algorithm is implemented to solve the resulting problem.

A genetic algorithm for the linear bilevel problem in which both objective functions are linear and the common constraint region is a polyhedron is developed in [20], the algorithm aims to combine classical extreme point enumeration techniques with genetic search methods by associating chromosomes with extreme points of the polyhedron. In [23] a genetic algorithm for obtaining good quality configurations of the topological design of LANs is proposed, the solution method considers the Stackelberg equilibrium to solve the bilevel problem and deals with the fact that the follower's problem cannot be optimally solved in a straightforward manner.

In this section, a genetic algorithm is proposed for solving the capacitated facility location problem with preferences. It is important to highlight that the solutions considered in the genetic algorithm are the ones associated with the leader's decision. This is the typical way to design algorithms for bilevel problems due to the necessity of having the follower's optimal solution for each leader's decision.

Taking advantage of the structure of the lower level, which is a transportation problem with integer supply and integer demand, the transportation simplex algorithm is proposed for its resolution. Also, owing to the existence of capacity in the problem, multiple solutions for allocating the customers to their preferred facilities when a predefined number of facilities has been located may exist, and it is considered in the genetic algorithm. Due to the fact that it is assumed the optimistic approach, a lexicographical method that guarantees the best convenience for the leader with an optimal solution of the follower is applied. That is, once the leader makes a decision the follower reacts returning his best solution. Then, the problem of the leader is solved again but with an additional restriction, the equality of the objective function of the follower and the best solution of the follower obtained above. If the solution obtained by the leader is different, then there are multiple solutions and the new solution obtained is taken. As well, the genetic algorithm examines whether all open facilities in the solution are used, this is, if an open facility does not have assigned customers to it, it is closed and the corresponding fixed cost is not considered in the objective function of the upper level.

Two versions of the proposed genetic algorithm are implemented. Both versions are detailed explained below.

Coding of the solutions

It is important to mention that part of the genetic algorithm's efficiency relies in having a convenient solution coding. Therefore, the coding of a leader's solution consists in a binary vector of size n in which each position indicates the index of an opened facility. If the n-th position is 1, then the n-th facility is opened, otherwise the facility is closed.

Initial population

A biased random methodology is considered to generate the initial population. First, two random numbers are generated between 10 and 10,000, later n random numbers between those values are generated. Now, these values will be fictitious fixed costs and are denoted as a_i. Then, a chromosome is generated by solving the following problem:

$$\min_{y} \quad \sum_{i \in I} a_i y_i \tag{6.40}$$

$$\text{subject to:} \sum_{i \in I} q_i y_i \quad \geq \quad m \tag{6.41}$$

$$0 \leq y_i \leq \quad 1 \qquad \forall i \in I \tag{6.42}$$

Due to the nature of the variable y, the optimal solution of the problem may have fractional elements. These elements will be repaired doing 1 all values that are greater than 0. Once the open facilities have been identified, their preferences, capacities and distribution costs are considered and the lower level problem is solved. After the fitness of the chromosome (solution) is evaluated by calculating the value of the objective function of the upper level, the initial population is formed to have P_{size} chromosomes

within the initial population (Pob).

Genetic operators

Genetic operators contribute to obtaining new solutions thus favouring the exploration of the search space. The basic operators used are: mutation and crossover. The mutation operator will give a new different solution (offspring) which is not very different from the selected solution. On the other hand, crossover will generate two new solutions (offspring) that inherit some characteristics from the two selected initial solutions (parents). The main contribution of crossover is expanding in the search space of possible solutions and the objective of the mutation is the introduction of randomness in the chromosomes of the population.

Crossover 1:
The strategy used is the single-point crossover. Each solution of Pob is paired with another randomly selected solution (parent solutions), and then a random crossover point is selected. After that, the parts of both solutions are exchanged, that is, the part of the solution that is left to the crossover point with the part of the other solution that is right to the crossover point. In this way the offspring inherit characteristics of both parents.

Crossover 2:
In this the crossover operator a procedure is performed to create a specific probability distribution by weighting the associated capacity with the open facilities in parent solutions. The main idea is to assign to each of these facilities a probability proportional to its capacity. In this procedure, each solution of the set Pob is paired with another solution randomly selected (parents solutions) of Pob. Later, the capacity of each open facility on the chromosome is identified, this calculation is done for both parents. If a facility is open in both parents, then the capacity is doubled; in this way that facility will have higher probability for staying opened. Therefore, a radius is calculated for all the facilities as follows:

$$r_i = \frac{q_i^1 + q_i^2}{\sum_{i \in I_p}(q_i^1 + q_i^2)} \tag{6.43}$$

where q_i^1 and q_i^2 denote the capacities of the facility of the parent 1 and parent 2, respectively. Also, let I_p indicates the open facilities in the parents solutions.

If the radius is equal to zero for any facility, then this facility will not be considered as a candidate because $r_i = 0$ indicates that the facility i is closed in both parents.

Finally, a probability to each facility is assigned. The idea is to divide the interval $[0, 1]$ in subintervals of length r_i, that is, assign to each open facility a subinterval. Then, a random number between 0 and 1 is generated and the facility that corresponds to that subinterval is opened. The number of facilities that are opened in the offspring is the number corresponding to the lower cardinality between both parents. In case when the offspring is infeasible due to capacity constraints, it will enter to the repairing scheme.

Mutation 1:
A solution is selected from set Pob. After that, a facility is randomly chosen from the solution and its value is modified (from 0 to 1 or from 1 to 0).

Mutation 2:
A solution is selected from set Pob. After that, two random numbers are generated: one that corresponds to an open facility and other one to a closed facility. Finally, both facilities are exchanged and an offspring is obtained.

Repairing scheme

Taking into account the existence of capacity constraints the offspring generated can be infeasible. Then these solutions are repaired as follows:

Repairing scheme 1:
First, one facility from the facilities that are not in the current solution is randomly selected. Then, this facility is opened in the current solution. If the solution continuous being infeasible then the procedure is repeated until it changes its status.

Repairing scheme 2:
First, from the facilities that are not opened in the current solution, the one that has more capacity is found, and this facility is opened in the offspring. If the offspring continuous being infeasible then the procedure is repeated until it changes its status.

Selection mechanism

The selection mechanism implemented is an elitist one, aiming to keep the best chromosomes. The solutions are ordered according to their objective function value and the best P_{size} solutions are selected.

The differences between the two versions implemented of the proposed genetic algorithm are the genetic operators and the repairing scheme. We denote as *genetic 0* to the algorithm that contains crossover 1, mutation 1 and repairing scheme 1 within its implementation. Therefore *genetic 1* is the one that uses crossover 2, mutation 2 and repairing scheme 2. In the next section the results obtained from both versions are presented.

6.1.4 Computational experimentation

The considered set of instances was taken from the instances generated in [69]. In that paper, the procedure with the 71 problems were generated is explained. These instances have to be modified to obtain facilities' capacity in terms of the number of customers who can be served from it. We compute $\bar{d}$ as the mean of the customer demands. Let be Q_i the capacity of the facility i, then $q_i = \lceil \frac{Q_i}{\bar{d}} \rceil$. The preferences are generated in a similar way than in [26]. The size of the instances is described below.

Table 6.1: Number of facilities and customers

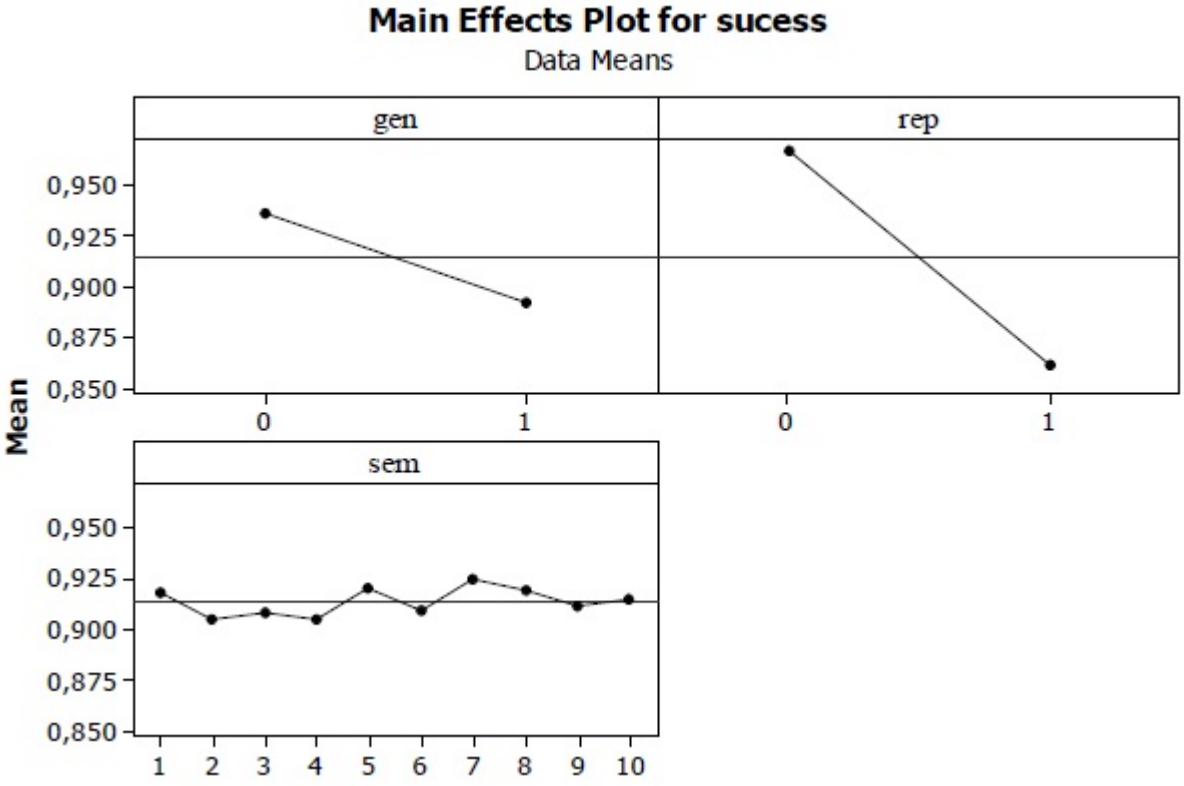
Instance	Facilities	Customers
p01-p12	10	50
p13-p24	20	50
p25-p40	30	150
p41-p55	10-30	70-100
p56-p71	30	200

The computational experimentation was conducted in a workstation with an Intel (R) Core processor and 32GB of RAM. The code was implemented in C++ language and CPLEX 12.6.1 was used for solving the reformulation.

Both versions of the genetic algorithm described above are analyzed. Each problem has been solved with each genetic algorithm (0 and 1) allowing (rep=1) or not (rep=0) repeating individuals, with 10 initial populations (sem) and 10 replicates each (run). We save the value of the solution after 1, 5, 10 and 30 seconds.

First, the genetic algorithms and repetition of chromosomes are analyzed (see Figure 6.1). In the Figure 6.2 it can be seen the interaction between the type of genetic, the repetition of chromosomes and the seed population.

Figure 6.1: Main effects plot for sucess.



Note that the *genetic 0* and the non-repetition of chromosomes have greater effect. From the above graphs, we concluded that we will not continue considering the repetition of chromosomes, because in general, for any problem and genetic algorithm the results are worse. In some particular problems, the behavior was different from the other ones. Hence, only the results for the first 40 problems are shown, see Figure 6.3.

From the above figure, it can be concluded that the success is greater when the chromosomes are not repeated. Then, repeating chromosomes is discarded and all the following results are not allowing repetition. Also, the number of iterations performed on each problem is analyzed, see Figures 6.4 and 6.5.

Figure 6.2: Interaction plot for sucess.

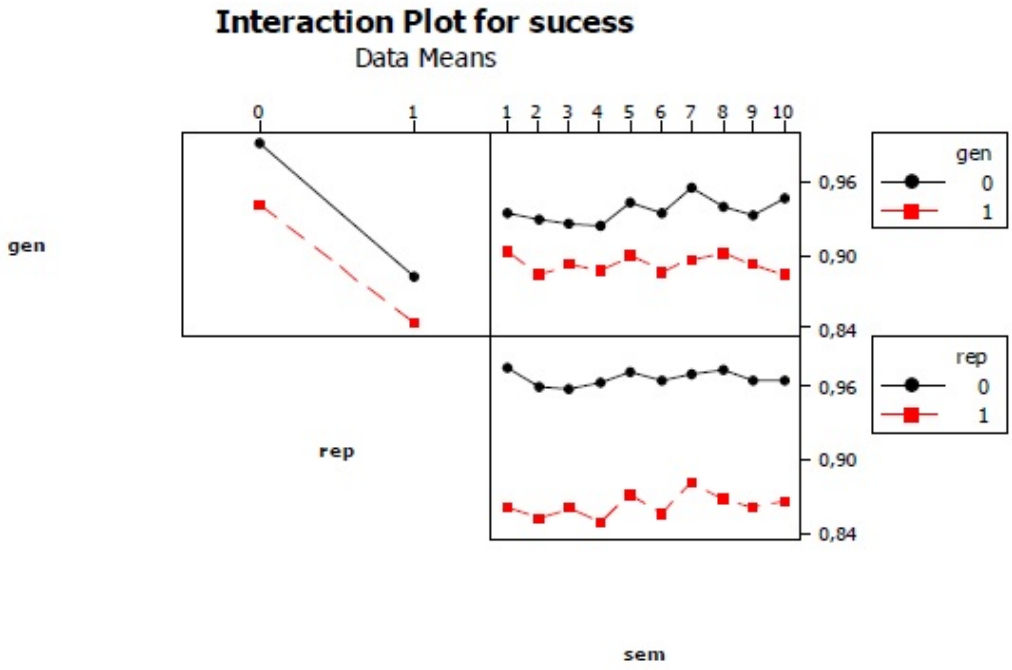
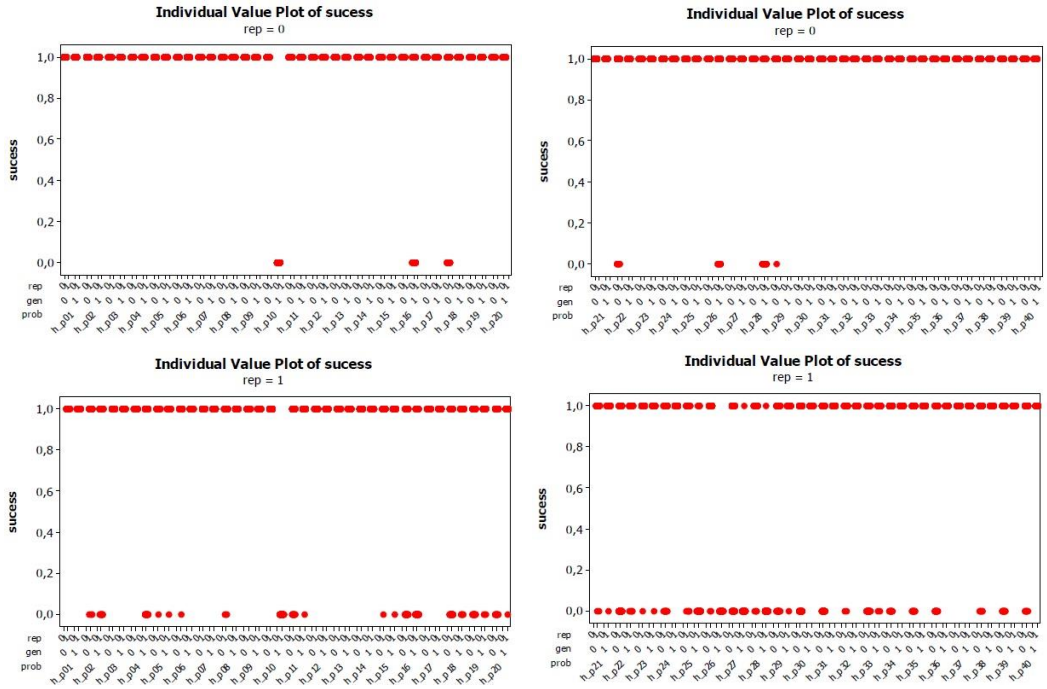
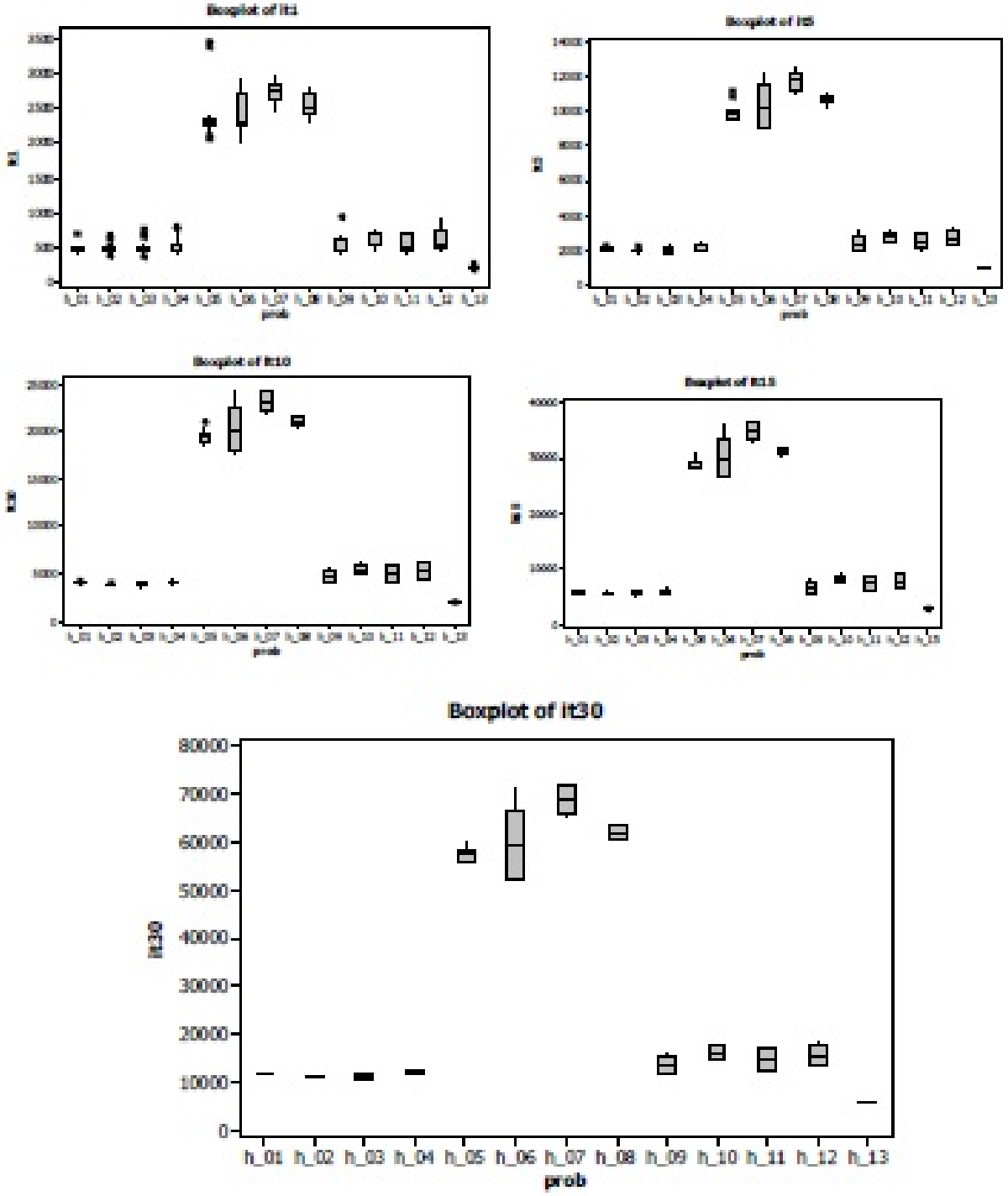


Figure 6.3: Individual value plot of success with rep=0 and rep=1.



Now, analyzing the number of iteration in which the best value was reached, it can be seen that this occurs realtively soon in almost all problems (see Figure 6.6).

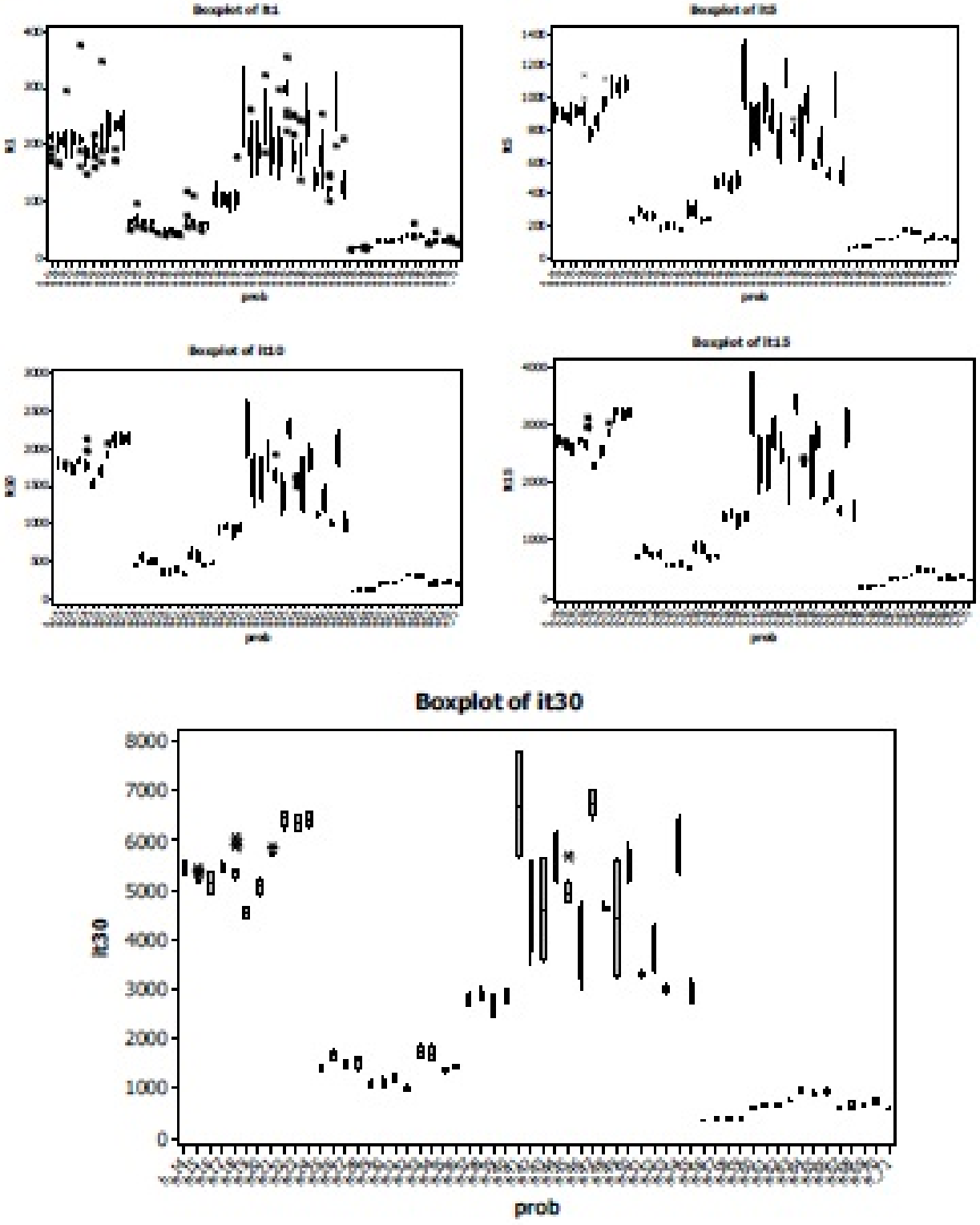
Figure 6.4: Number of iterations in problems h_p01 to h_p13.



Finally, the success rate for each genetic algorithm applied to each problem after n seconds is presented in Figure 6.7. A success is defined as achieving the best value.

Therefore, more statistical tests analyzing both genetic algorithms are performed. First, the interaction, the effect, the independence among variables, the effect of each genetic, among others are analyzed.

Figure 6.5: Number of iterations in problems h_p14 to h_p71.



As a conclusion, both algorithms behave reasonably well, although the *genetic 0* is slightly better than the *genetic 1*, being less stuck in local optimum and reaching the optimum in less time and in more

Figure 6.6: Number of iteration in which the best value has been reached.

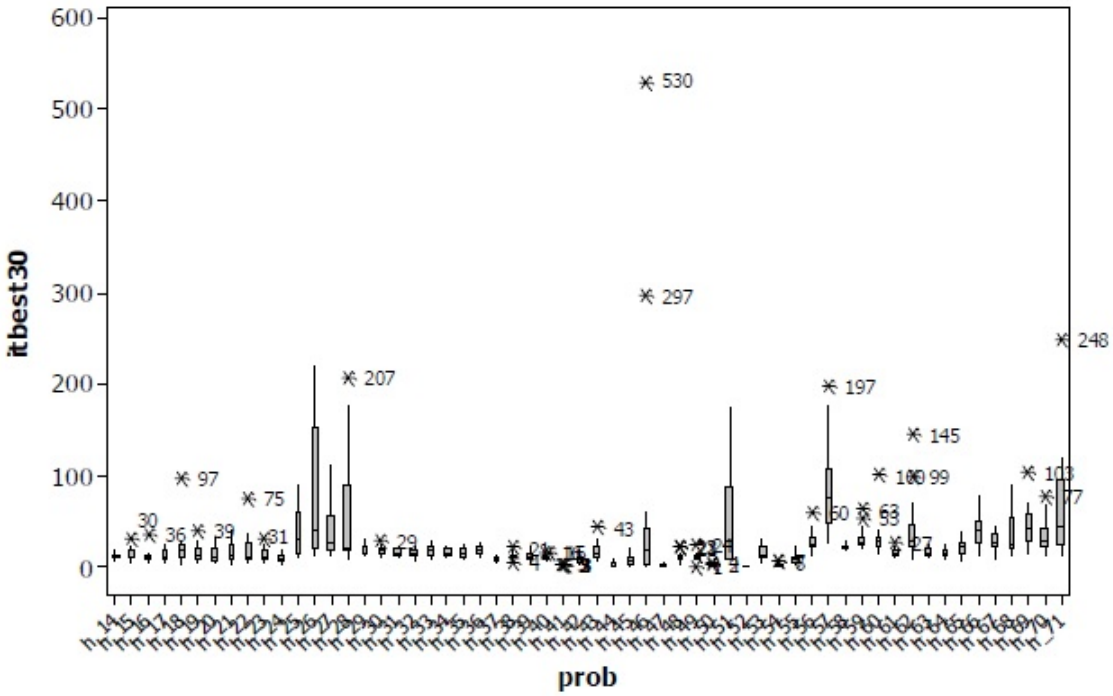
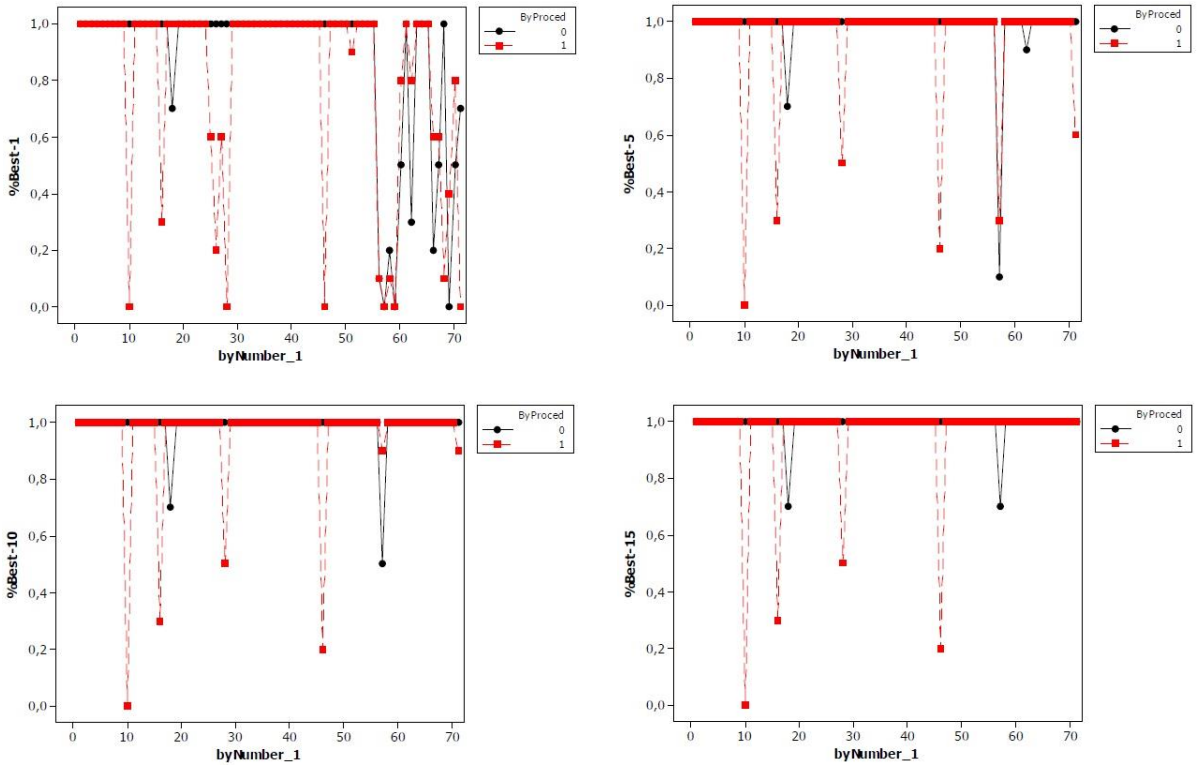


Figure 6.7: Variable success after 1,5,10 and 15 seconds.



problems. So, we decided to consider the genetic 0. for the final experimentation.

The following tables show the results obtained (6.2 and 6.3). The first column called “instance” corresponds to the instance label, the “Optimum / Best” and “Time” columns indicate the best value of the objective function and the time obtained with the reformulation presented in Section 6.1.2. A time limit of two hours were considered. On the other hand, due to the existence of randomness, the genetic algorithm was executed ten times for each problem. The last three column shows the average percentage deviation of the best solutions obtained in each instance with respect to the optimal or best known solution (Best), the average percentage deviation of the solution’s mean with respect to the optimal or best known solution (Average), and the average percentage deviation of the worst solutions obtained with respect to the optimal or best known solution (Worst).

From the above results, it can be appreciated that the best value found by CPLEX within the time limit coincides with that obtained by the genetic algorithm in instances p25 to p36 and p59 to p66. This seems to indicate that the value found by CPLEX may be optimal. On the other hand, the genetic algorithm found a better value of the objective function in the instances p56-p58 and p67-p69. Also, it is easy to see that in only three instances the GA did not find the optimum in the ten runs, in p18, p57 and p62. However, the algorithm obtained optimal solutions in 7 out of 10, 1 out of 10, and 9 out of the 10 runs, respectively. It should be noted that although in the instance p57 only found it in 1 out of 9 in the first five seconds, at the end of the 30 seconds had found the optimum in 7 out of 10 runs.

Table 6.2: Results obtained from the computational experimentation.

	Reformulation		**Genetic algorithm**		
Instance	**Optimum/Best**	**Time (second)**	**Best**	**Average**	**Worst**
p01	19902	0.34	0.0000	0.0000	0.0000
p02	19034	0.30	0.0000	0.0000	0.0000
p03	19151	0.33	0.0000	0.0000	0.0000
p04	21949	0.31	0.0000	0.0000	0.0000
p05	19519	0.04	0.0000	0.0000	0.0000
p06	19458	0.05	0.0000	0.0000	0.0000
p07	19456	0.06	0.0000	0.0000	0.0000
p08	21309	0.04	0.0000	0.0000	0.0000
p09	18794	0.61	0.0000	0.0000	0.0000
p10	16571	0.30	0.0000	0.0000	0.0000
p11	19048	0.34	0.0000	0.0000	0.0000
p12	18544	0.39	0.0000	0.0000	0.0000
p13	17723	94.44	0.0000	0.0000	0.0000
p14	16139	65.02	0.0000	0.0000	0.0000
p15	18055	125.39	0.0000	0.0000	0.0000
p16	19206	142.81	0.0000	0.0000	0.0000
p17	17234	71.02	0.0000	0.0000	0.0000
p18	16349	58.55	0.0000	0.0030	0.0099
p19	18115	99.66	0.0000	0.0000	0.0000
p20	19883	111.64	0.0000	0.0000	0.0000
p21	17253	47.22	0.0000	0.0000	0.0000
p22	16415	54.47	0.0000	0.0000	0.0000
p23	18146	39.94	0.0000	0.0000	0.0000
p24	18321	79.78	0.0000	0.0000	0.0000
p25	40164	7200.00	0.0000	0.0000	0.0000
p26	43187	7200.00	0.0000	0.0000	0.0000
p27	41500	7200.00	0.0000	0.0000	0.0000
p28	45474	7200.00	0.0000	0.0000	0.0000
p29	45026	7200.00	0.0000	0.0000	0.0000
p30	44075	7200.00	0.0000	0.0000	0.0000
p31	42246	7200.00	0.0000	0.0000	0.0000
p32	48460	7200.00	0.0000	0.0000	0.0000
p33	42184	7200.00	0.0000	0.0000	0.0000
p34	38846	7200.00	0.0000	0.0000	0.0000
p35	40657	7200.00	0.0000	0.0000	0.0000
p36	45392	7200.00	0.0000	0.0000	0.0000

Table 6.3: Results obtained from the computational experimentation (continuation).

	Reformulation		Genetic algorithm		
Instance	**Optimum/Best**	**Time (second)**	**Best**	**Average**	**Worst**
p37	35593	1654.36	0.0000	0.0000	0.0000
p38	35355	1354.81	0.0000	0.0000	0.0000
p39	35642	386.00	0.0000	0.0000	0.0000
p40	36354	391.16	0.0000	0.0000	0.0000
p41	11574	0.36	0.0000	0.0000	0.0000
p42	9708	30.63	0.0000	0.0000	0.0000
p43	8637	1007.52	0.0000	0.0000	0.0000
p44	16426	0.52	0.0000	0.0000	0.0000
p45	12514	56.52	0.0000	0.0000	0.0000
p46	10741	1412.67	0.0000	0.0000	0.0000
p47	13534	0.42	0.0000	0.0000	0.0000
p48	11070	40.22	0.0000	0.0000	0.0000
p49	9175	1120.75	0.0000	0.0000	0.0000
p50	16749	0.44	0.0000	0.0000	0.0000
p51	15510	108.08	0.0000	0.0000	0.0000
p52	21872	0.38	0.0000	0.0000	0.0000
p53	20358	140.14	0.0000	0.0000	0.0000
p54	19114	0.34	0.0000	0.0000	0.0000
p55	17405	117.67	0.0000	0.0000	0.0000
p56	64807	7200.00	-0.0003	-0.0003	-0.0003
p57	70253	7200.00	-0.0002	-0.0001	-0.0001
p58	81921	7200.00	-0.0024	-0.0024	-0.0024
p59	75080	7200.00	0.0000	0.0000	0.0000
p60	61711	7200.00	0.0000	0.0000	0.0000
p61	62790	7200.00	0.0000	0.0000	0.0000
p62	72339	7200.00	0.0000	0.0005	0.0051
p63	66531	7200.00	0.0000	0.0000	0.0000
p64	60709	7200.00	0.0000	0.0000	0.0000
p65	63290	7200.00	0.0000	0.0000	0.0000
p66	69953	7200.00	0.0000	0.0000	0.0000
p67	70125	7200.00	-0.0107	-0.0107	-0.0107
p68	62665	7200.00	-0.0053	-0.0053	-0.0053
p69	67509	7200.00	-0.0072	-0.0072	-0.0072
p70	71740	7200.00	0.0000	0.0000	0.0000
p71	66950	7200.00	0.0000	0.0000	0.0000

6.2 Capacitated facility location problem with generalized demand

None of the papers mentioned in the literature review present an appropriate definition of the concepts for approximating the inducible region or for handling solutions that do not belong to the bilevel feasible region. We define *attainable bilevel solutions* based on an efficient approximation of the inducible region. Furthermore, heuristic algorithms are developed for solving the bilevel capacitated facility location problem. The impact of having refined methods for obtaining the follower's best acceptable response is shown. Hence, an alternative for solving this complex problem is proposed within this section.

6.2.1 Bilevel model description

Let $I = \{1, 2, ..., n\}$ be the set of potential location for the facilities, $J = \{1, 2, ..., m\}$ be the set of customers served by the facilities. The parameters, c_{ij} be the associated costs for allocating customer j to facility i and f_i be the fixed cost for opening a facility in the potential location i. Also, let d_j be the demand associated with each customer j and b_i the capacity for each facility i. As in previous chapters, let g_{ij} be the preference of customer j toward facility i.

Let also define the variables:

$$y_i = \begin{cases} 1, & \text{if a facility is located in potential location } i \\ 0, & \text{otherwise} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1, & \text{if a facility } i \text{ supplies the demand of customer } j \\ 0, & \text{otherwise} \end{cases}$$

The mathematical model for the Bilevel Capacitated Facility Location Problem (BCFLP) considering the preferences of the customers towards the facilities is set as follows:

$$\min_{y,x} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{6.44}$$

$$\text{subject to: } y_i \in \{0,1\} \qquad \forall i \in I \tag{6.45}$$

$$x \in \arg\max \quad \sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} \tag{6.46}$$

$$\text{subject to: } \sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{6.47}$$

$$\sum_{j\in J} d_j x_{ij} \leq b_i y_i \qquad \forall i \in I \tag{6.48}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{6.49}$$

Equation (6.44) denotes the leader's objective function, which minimizes locating and distributing costs. In (6.45), the binary nature of the leader's decision variables is stated. Constraint (6.46) is the follower's objective function which maximizes the preferences set by the customers. The requirement that a customer's whole demand must be met is guaranteed by equation (6.47). The capacity constraint

for the facilities is considered in (6.48), where the located facilities will supply the demand of all the possible customers without exceeding their production capacity. In (6.49), a single facility will supply the demand of a customer. In the BCFLP, the leader has control over the decision variables y_i; that is, the leader will choose where potential facilities will be located. On the other hand, the follower controls the decision variables x_{ij}; that is, the follower will allocate the customers to the most convenient facilities from a subset (decided by the leader) of located ones.

As in the previous section, to have well-posed the bilevel problem studied in this section, the optimistic or pessimistic approach must be specified. Both decision variables appearing in the leader's objective function imply the classical optimistic approach. This approach can be seen from two different perspectives: (1) the follower has a cooperative behavior or (2) the leader selects the most favorable follower's decision with respect to its own objective value, being aware the follower's objective function value remains the same.

6.2.2 Concepts and terminology

The following concepts are used in [99] and [11] for problems where the lower level can be solved to optimality. Hence, the necessary concepts for having well-posed the bilevel problem defined in the previous section are introduced. Because the lower level problem of the BCFLP is NP-hard, the bilevel feasibility for the proposed solutions is defined as follows:

Definition 1. Let $x = (x_{11}, x_{12}, ..., x_{1m}, x_{21}, x_{22}, ..., x_{2m}, ..., x_{n1}, x_{n2}, ..., x_{nm})$ be the decomposed vector which indicates the customer's demand covered by the facilities. Also, let $y =(y_1, y_2, ..., y_n)$ be the vector of the potential located facilities. Then, the constraint set of the BCFLP is:

$$S = \{(x, y) : y_i \in \{0, 1\}, \sum_{i \in I} x_{ij} = 1, \sum_{j \in J} d_j x_{ij} \leq b_i y_i \quad and \quad x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J\} \tag{6.50}$$

Definition 2. For a given leader's decision y fixed, constraints (6.47)-(6.49) define the follower's feasible set $S(y)$ as follows:

$$S(y) = \{x : \sum_{i \in I} x_{ij} = 1, \sum_{j \in J} d_j x_{ij} \leq b_i y_i \quad and \quad x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J\} \tag{6.51}$$

The complexity required to solve a general case of the lower level is evident, it is a NP-hard problem. Hence, obtaining optimal solutions is not guaranteed for all instances. Moreover, applying refined numerical methods for solving the follower's problem will negatively affect the performance of any algorithm developed for solving the bilevel problem. Due to these facts, a definition of a set of bounds for any leader's decision y is introduced.

Definition 3. Let $\epsilon > 0$ be a tolerance value. Hence, the bounds set $B(y)$ can be defined as:

$$B(y) = \{x : x \in S(y), \quad and \quad \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} + \epsilon = \sum_{i \in I} \sum_{j \in J} g_{ij} x^*_{ij}\} \tag{6.52}$$

In a general way, the optimal solutions of the lower level problem cannot be obtained by an exact method. Furthermore, it can be seen from the definition stated above that $B(y)$ contains solutions yielding to lower bounds. The configuration of the lower level solution affects the quality of the upper level's objective function in a non-predictable way. The importance of having good quality solutions will affect the efficiency of $B(y)$ and, consequently, the bilevel attainability of solutions. In other

words, near optimal solutions x will be desired to improve the $B(y)$ set and to maintain it very close to the inducible region associated with the integer bilevel problem; that is, when ϵ has a value close to zero.

Follower's rational reaction set must be adjusted for solving the BCFLP. The solution x may be seen as the follower's rational reaction for a fixed leader's decision y. By rational, we mean the follower will react in a logical sense; that is, if optimal solutions are very difficult or time-consuming to obtain, the follower will accept near optimal solutions as his reaction.

Definition 4. The rational reaction set $P(y)$ may be defined as:

$$P(y) = \{x : x \in B(y)\} \tag{6.53}$$

Definition 5. The approximate inducible region IR considered for the BCFLP is defined by the set:

$$IR = \{(x, y) : (x, y) \in S \quad and \quad x \in B(y)\} \tag{6.54}$$

From the latter definition, the approximate inducible region will represent the approximate region at the leader's problem. The points belonging to the defined approximate inducible region will be referred to as the bilevel attainable solutions.

Definition 6. A solution (x, y) will be a bilevel attainable solution if $(x, y) \in IR$.

As mentioned before, the optimal reaction of the follower cannot be assured since the lower level problem is NP-hard. In any event, we seek to obtain the best bilevel attainable solution -a solution $(\hat{x}, \hat{y}) \in IR$ corresponding to the minimum objective function value reached, according to:

$$\sum_{i \in I} \sum_{j \in J} c_{ij} \hat{x}_{ij} + \sum_{i \in I} f_i \hat{y}_i \tag{6.55}$$

Hence, the need of considering efficient methods for obtaining solutions in the lower level problem is evident. In Section 6.2.6, computational results from different strategies for solving the lower level (obtaining bounds) are presented. Finally, as mentioned in the previous section, an optimistic approach is assumed. In other words, in the case when $P(y)$ is not a singleton (if multiple optimal solutions for the lower level problem exist), then the follower selects the solution that corresponds to the best leader's objective function value.

6.2.3 A bound for the BCFLP

In this section, a bound for the bilevel capacitated facility location problem is presented. The common bound for bilevel programming problems consists in ignoring the follower's objective function and solving the resulting problem. Since we are considering a minimization problem, the resulting bound under that scheme is a lower one. By doing so, the upper level's decision maker will decide both decision variables. Hence, he will select the best decisions based on his own objective function without regarding the lower level.

Therefore, the single-level reformulation used for obtaining classical lower bounds of the problem (6.44)-(6.49) is as follows:

$$\min_{y,x} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{6.56}$$

$$\text{subject to: } y_i \in \{0,1\} \qquad \forall i \in I \tag{6.57}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{6.58}$$

$$\sum_{j\in J} d_j x_{ij} \leq b_i y_i \qquad \forall i \in I \tag{6.59}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{6.60}$$

This scheme is very useful for validate the bilevel structure of a problem. This is, if the optimal solution of the single-level problem (6.56)-(6.60) is the same than the optimal solution of the bilevel problem (6.44)-(6.49), then it implies that the most preferred facilities coincide with the closest ones; and the bilevel formulation lost it sense. Moreover, since the decision of the lower level decision maker is not being taken into account, the upper level objective function cannot be affected in any way. Hence, the optimal solution of the problem (6.56)-(6.60) will be a lower bound for the bilevel problem.

However, very poor quality bounds may be obtained. Hence, a different analysis to obtain bounds is conducted in this thesis. The proposed bound is based on a reformulation of the BCFLP's LP-relaxation, into a single-level mixed integer programming problem. Since the follower's problem is an integer maximization problem, its LP-relaxation will result in an upper bound for the follower's problem. Then, one would expect the inducible region to be overestimated and the minimization over that region to be a upper bound for the bilevel problem. In general, this is not true, however, since the leader's objective function partially depends on the solution given by the follower. Moreover, the inducible region cannot be predicted due to the appearance of solutions that overestimate or underestimate the objective function value without a pattern. In Section 6.2.4, an explanation of this fact is discussed, and an illustrative example is presented.

As stated above, to obtain a bound, a relaxation of the binary constraints of the follower's problem is considered -that is, substitute $x_{ij} \in \{0,1\}$ by $x_{ij} \geq 0$. By doing this, the resulting lower level problem is a linear programing problem. Hence, the well-known reformulation for bilevel programming problems in which the lower level problem is replaced by its primal-dual constraints is developed. Let u_j and v_i be the dual variables associated with the follower's problem.

Therefore, the dual formulation of the follower's problem is:

$$\min_{u,v} \quad \sum_{j\in J} u_j + \sum_{i\in I} b_i y_i v_i \tag{6.61}$$

$$\text{subject to: } u_j + d_j v_i \geq g_{ij} \qquad \forall i \in I, j \in J \tag{6.62}$$

$$v_i \geq 0 \qquad \forall i \in I \tag{6.63}$$

To guarantee the optimality of the lower level problem's relaxation, two frameworks are considered: establish the equality of both objective functions, primal and dual ones; or, include the complementary slackness constraints. First, for the reformulation R1, the constraint that ensures both objective

functions have the same value is amended in the problem. This constraint is as follows:

$$\sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} = \sum_{j\in J} u_j + \sum_{i\in I} b_i y_i v_i \tag{6.64}$$

Equation (6.64) is nonlinear, but it can be linearized in a classical manner by introducing the artificial variable z_i, and then making the substitution $z_i = y_i v_i$. If $y_i = 0$ then $z_i = 0$; and if $y_i = 1$ then $z_i = v_i$. This can be done by adding the following constraints, in which M_1 is a sufficiently large positive constant:

$$\begin{aligned} z_i &\geq 0 \qquad \forall i \in I && (6.65)\\ z_i &\leq M_1 y_i \qquad \forall i \in I && (6.66)\\ z_i &\leq v_i \qquad \forall i \in I && (6.67)\\ z_i - M_1 y_i &\geq v_i - M_1 \qquad \forall i \in I && (6.68) \end{aligned}$$

Then, the equation that will ensure the optimality is given by:

$$\sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} = \sum_{j\in J} u_j + \sum_{i\in I} b_i z_i \tag{6.69}$$

The first reformulation R1 of the relaxed BCFLP results in a mixed integer programming problem, which is composed of the following equations:

$$\min_{y,x,u,v,z} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{6.70}$$

$$\begin{aligned} \text{subject to: } \sum_{i\in I} x_{ij} &= 1 \qquad \forall j \in J && (6.71)\\ \sum_{j\in J} d_j x_{ij} &\leq b_i y_i \qquad \forall i \in I && (6.72)\\ u_j + d_j v_i &\geq g_{ij} \qquad \forall i \in I, j \in J && (6.73)\\ \sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} &= \sum_{j\in J} u_j + \sum_{i\in I} b_i z_i && (6.74)\\ z_i &\geq 0 \qquad \forall i \in I && (6.75)\\ z_i &\leq M_1 y_i \qquad \forall i \in I && (6.76)\\ z_i &\leq v_i \qquad \forall i \in I && (6.77)\\ z_i - M_1 y_i &\geq v_i - M_1 \qquad \forall i \in I && (6.78)\\ y_i &\in \{0,1\} \qquad \forall i \in I && (6.79)\\ x_{ij} &\geq 0 \qquad \forall i \in I, j \in J && (6.80)\\ v_i &\geq 0 \qquad \forall i \in I && (6.81) \end{aligned}$$

The second framework mentioned above that guarantees the optimality of the relaxed lower level problem is described. This second reformulation R2 is based on the complementary slackness constraints. For the follower's problem under study, these equations are as follows:

$$x_{ij}(u_j + d_j v_i - g_{ij}) = 0 \qquad \forall i \in I, j \in J \tag{6.82}$$

$$v_i(b_i y_i - \sum_{j \in J} d_j x_{ij}) = 0 \qquad \forall i \in I \tag{6.83}$$

Similarly equations (6.82) and (6.83) are nonlinear and a linearization is needed to improve the treatability of the problem. Let M_2 be a sufficiently large positive constant. First, consider equation (6.82) to linearize it. Remember that constraint (6.62) and that x_{ij} demands non-negativity. The introduction of auxiliary variables $\alpha_{ij} \in \{0, 1\}$ is made, including the following constraints:

$$x_{ij} \leq M_2(1 - \alpha_{ij}) \qquad \forall i \in I, j \in J \tag{6.84}$$

$$u_j + d_j v_i - g_{ij} \leq M_2 \alpha_{ij} \qquad \forall i \in I, j \in J \tag{6.85}$$

For linearizing equation (6.83), a similar procedure to the one formerly described can be conducted. That is, considering another constant M_3, non negativity of v_i, constraint (6.48) and auxiliary variables $\beta_i \in \{0, 1\}$ the resulting constraints will be added in the model:

$$b_i y_i - \sum_{j \in J} d_j x_{ij} \leq M_3 \beta_i \qquad \forall i \in I \tag{6.86}$$

$$v_i \leq M_3(1 - \beta_i) \qquad \forall i \in I \tag{6.87}$$

The resulting linearized single-level mixed integer programming problem R2 is:

$$\min_{y,x,u,v,\alpha,\beta} \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{6.88}$$

$$\text{subject to: } \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J \tag{6.89}$$

$$\sum_{j \in J} d_j x_{ij} \leq b_i y_i \qquad \forall i \in I \tag{6.90}$$

$$u_j + d_j v_i \geq g_{ij} \qquad \forall i \in I, j \in J \tag{6.91}$$

$$x_{ij} \leq M_2(1 - \alpha_{ij}) \qquad \forall i \in I, j \in J \tag{6.92}$$

$$u_j + d_j v_i - g_{ij} \leq M_2 \alpha_{ij} \qquad \forall i \in I, j \in J \tag{6.93}$$

$$b_i y_i - \sum_{j \in J} d_j x_{ij} \leq M_3 \beta_i \qquad \forall i \in I \tag{6.94}$$

$$v_i \leq M_3(1 - \beta_i) \qquad \forall i \in I \tag{6.95}$$

$$y_i \in \{0, 1\} \qquad \forall i \in I \tag{6.96}$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J \tag{6.97}$$

$$v_i \geq 0 \qquad \forall i \in I \tag{6.98}$$

$$\alpha_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{6.99}$$

$$\beta_i \in \{0, 1\} \qquad \forall i \in I \tag{6.100}$$

Without considering the variables' sign constraints, R1 model contains $mn + 3n + 1$ additional functional constraints and $m + 2n$ extra variables. On the other hand, $5mn + 2n$ new constraints and $mn + m + 2n$ more variables are added in R2. Therefore, the R2 model will always be larger than the R1 model in terms of the number of constraints and variables, even in the smallest case.

Considering both reformulations of the relaxed BCFLP are equivalent, and based on the fact R2 has a larger number of variables and constraints, solely the R1 model was considered to obtain bounds for the BCFLP. If the number of customers is larger than the number of facilities, then the optimality gap may be small (see [28]). On the other hand, a methodology for evaluating the quality of the bilevel solution obtained by the relaxed R1 model has not been proposed in the literature. Despite this, solutions provided by both reformulations are bounds for the BCFLP.

6.2.4 Analyzing the complexity in obtaining good quality bounds on the BCFLP

In this section, a discussion about the efficiency of obtaining lower or upper bounds for the BCFLP is presented. As mentioned in Section 6.2.2, the lower level is determined by the well-known GAP, a NP-hard problem. Hence, finding the optimal solution for the follower is impossible in an efficient and general way. The bilevel problem requires the optimal follower's solution, however. Knowing the limitations the follower has to achieve this, a rational response can be allowed for having bilevel attainable solutions.

Since bilevel attainable solutions are not bilevel feasible (in the strict sense), one assumes bounds exist for the BCFLP. Nevertheless, there is no guarantee the bilevel attainable solutions are upper or lower bounds. Actually, the methodology described in section 6.2.3 cannot assure the obtained solutions are upper or lower bounds. This shows the enormous difficulty in solving bilevel problems, especially those with a NP-hard problem in one of its two hierarchized decision levels.

Regarding the condition of the solutions obtained from relaxing the lower level problem (see section 6.2.3), first, denote the leader's objective function given in equation (6.44) as $F(x, y)$ and the follower's objective function defined in equation (6.46) as $f(x, y)$. Also, let y be a fixed leader's decision. Considering the follower's problem and since the linear relaxation of the follower's variables x_{LP} is considered and that it is a maximization problem, a greater or equal value of the follower's objective function is expected, $f(x^*_{LP}, y) \geq f(x, y)$ for any feasible solution $x \in \{0, 1\}$, where x^* and x^*_{LP} are the optimal solutions of the lower level, integer and relaxed respectively. Then, an approximation of the inducible region (IR_{LP}) is formed by all the pairs of solutions (x^*_{LP}, y) obtained through this linear relaxation. Therefore, one would also expect the solutions belonging to IR_{LP} overestimate (in terms of the leader's objective function value) the solutions in the inducible region (IR) of the integer bilevel problem, $F(x^*_{LP}, y) \geq F(x^*, y)$. This is not true for all the solutions, however, because some $(x^*, y) \in IR$ satisfy $F(x^*, y) > F(x^*_{LP}, y)$. Due to this fact, it cannot be assured that upper bounds for the BCFLP are obtained through the lower level's linear relaxation.

To illustrate this finding, a small example is shown. The problem consists of 7 facilities and 4 customers, and the parameters of the problem are presented in Figure 6.8. For example, consider the case when the leader locates the first and fourth facilities, that is, $y = (1, 0, 0, 1, 0, 0, 0)$. Hence, the

optimal follower's response (in a coded way for simplicity) is $x^* = \{x_{11} = x_{14} = x_{42} = x_{43} = 1\}$ with $f(x^*, y) = 20$. On the other hand, the optimal solution for the linear relaxation of the follower's problem is $x^*_{LP} = \{x_{11} = x_{42} = x_{43} = 1, x_{14} = 0.122, x_{44} = 0.878\}$ with $f(x^*_{LP}, y) = 20.878$. Therefore, computing the corresponding leader's objective function values results in $F(x^*, y) = 285$ and $F(x^*_{LP}, y) = 280.610$; that is, the bilevel solution given by the linear relaxation of the lower level problem does not yield to a upper bound. Not all the solutions have the same behavior -there are some bilevel solutions that satisfy $F(x^*_{LP}, y) \geq F(x^*, y)$.

Figure 6.8: Parameters for the illustrative example.

					f_i	b_i				
	5	1	6	10	130	213	6	1	3	4
	3	4	4	6	77	141	3	3	6	6
	5	3	3	9	104	158	5	7	2	7
c_{ij}	4	1	5	5	134	186	4	5	5	5	g_{ij}
	7	5	2	8	73	176	1	2	1	1
	6	1	6	5	177	208	2	4	4	3
	5	6	5	2	93	197	7	6	7	2
d_j	77	59	62	74						

In an analogous way, one can analyze the case for the supposed lower bounds. For a given leader's decision y, if the lower level problem is solved by a heuristic algorithm, there is no guarantee the obtained solutions x_{Heu} are optimal. Additionally, since the follower aims to maximize, a pair of solutions (x_{Heu}, y) should satisfy $f(x_{Heu}, y) \leq f(x^*, y)$ for the optimal follower's solution x^*, yielding to a lower bound. Then, the inducible region is underestimated, which would suggest that $F(x_{Heu}, y) \leq F(x^*, y)$. As in the case for the upper bounds, this is not true for all the solutions, however.

Considering again the example presented in Figure 6.8, suppose that the leader has made a decision $y = (0, 0, 0, 1, 0, 0, 1)$. For the integer case, the follower's optimal solution is $x^* = \{x_{42} = x_{44} = x_{71} = x_{73} = 1\}$ with $f(x^*, y) = 24$. Now, consider the lower level problem is solved through a heuristic algorithm. Based on the criteria selected for the heuristic, the follower's rational response is $x_{Heu} = \{x_{43} = x_{44} = x_{71} = x_{72} = 1\}$ with $f(x_{Heu}, y) = 23$. Finally, from evaluating those solutions in the leader's objective function we will have $F(x^*, y) = 243$ and $F(x_{Heu}, y) = 248$. This result shows that solving by a heuristic algorithm the lower level may result in leader's objective function values greater than the ones obtained in an optimal manner.

After having analyzed this small example, one notes the validation of the lower or upper bounds for the BCFLP cannot be done. In the case when the linear relaxation is made, one would expect to obtain upper bounds, but this is not always correct. Also, when a heuristic algorithm is used for solving the lower level problem, the finding of lower bounds would be expected. Again, this is not true in a general way. Moreover, under both approaches, there will be solutions that become an overestimation or underestimation of the bilevel optimal solution. Despite the impossibility of making claims about the classification of bounds, the BCFLP needs to be solved in an efficient way. The results presented

in Section 6.2.6 show the inefficiency of solving the BCFLP by using the reformulation R1 for all instances. Therefore, the design of a refined heuristic algorithm, able to solve large-size instances of the problem in an efficient manner, is needed. The proposed algorithm is based in the cross entropy metaheuristic for seeking leader's solutions and uses three different schemes for solving the follower's problem -the detailed description is presented in the next section.

6.2.5 A cross entropy method for the BCFLP

In this section, three solution procedures are proposed. The first obtains bilevel feasible solutions, while the other two obtain bilevel attainable ones. The main differences between the three proposed procedures are based on the strategy for solving the GAP that appears in the lower level. For dealing with the upper level problem, a Cross Entropy heuristic is developed.

6.2.5.1 Cross Entropy method

Cross Entropy (CE) (see [35], [79], [119]) is a method developed for the estimation of rare-event probabilities. Then, due to its potential, CE is adapted for solving combinatorial optimization problems. CE is a two-phase iterative method in which a random sample is generated by a specified mechanism; then, the parameters of the generator mechanism are updated considering the data sample results to improve the sample in the next iteration.

Within the CE method, an efficient or optimal solution for the lower level problem must be obtained. To achieve this, three alternatives are considered. The first two methods follow GRASP principles; one constructs solutions based only on customer's preferences, and the second one considers a generalized regret measure. The third method uses CPLEX for getting the exact solution, in instances when this is possible.

GRASP is a multi-start procedure composed of two phases: (i) construction and (ii) improvement. In each iteration, GRASP looks forward to construct a solution in a greedy randomized adaptive fashion by using a Restricted Candidate List (RCL). Then, the current solution enters into the improvement phase for conducting a local search keeping the best solution found (see [50]). Within the proposed algorithm, two different construction methods that follow GRASP principles were developed for solving the lower level problem -customer-facility allocation. The two versions of construction methods differ from each other in the way the evaluation of a candidate's contribution is made -the criterion used for selecting the allocation when a solution is being created. The first version uses customer preferences, while in the second one, an adaptation of the generalized regret measure proposed in [105] is considered. Although the proposed construction methods employ a greedy randomized adaptive component during construction, they cannot be considered a GRASP because we do not consider the improvement phase (due to the increase in the required computational time); hence, the efficiency of the methods is diminished.

In Algorithm 6.2.1, a general pseudocode for the CE proposed is depicted. All the procedures mentioned in this procedure will be detailed later in this section.

Algorithm 6.2.1 Pseudocode of the CE method.

procedure CROSS ENTROPY(maxIter, N, d, λ)
  $k \leftarrow 1$
  $nbIter \leftarrow 0$
  $t \leftarrow 0$
  $p_i^t \leftarrow 0.5 \quad \forall i \in I$ (initial probability)
  $y\_incumbent \leftarrow \emptyset$ (incumbent leader's solution)
  $F^* \leftarrow \infty$ (incumbent leader's objective function value)
  while $(nbIter < maxIter)$ **do**
    while $(k \leq N)$ **do**
      $y \leftarrow Facilities\_Location(p^t)$
      $x \leftarrow Customers\_Allocation(y)$
      $F \leftarrow Evaluate\_UpperLevel(y, x)$
      $Sample\_obj(k) \leftarrow F$
      $Sample\_sol(k) \leftarrow y$
      $k \leftarrow k + 1$
    end while
    $Sample\_obj \leftarrow sort(Sample\_obj)$ (in increasing order)
    $Sample\_sol \leftarrow sort(Sample\_sol)$ (corresponding with $Sample\_obj$)
    $F\_incumbent \leftarrow Sample\_obj(1)$
    $y\_incumbent \leftarrow Sample\_sol(1)$
    $q_i = \sum_{k=1}^{n} y_i^k / d \quad \forall i \in I$ where d denotes the d-th first solutions of $Sample\_sol$
    $p_i^{(t+1)} = \lambda(q_i) + (1 - \lambda)p_i^t \quad \forall i \in I$
    if $(F\_incumbent < F^*)$ **then**
      $F^* \leftarrow F\_incumbent$
      $y^* \leftarrow y\_incumbent$
    end if
    $nbIter \leftarrow nbInter + 1$
    $t \leftarrow t + 1$
  end while
  return F^*, y^*
end procedure

As illustrated in the pseudocode in 6.2.1, for initializing the CE method a set of N leader's solutions needs to be constructed. For each of these solutions, the lower level problem is solved and evaluated. This information is needed for selecting a sample of size d ($d < N$), which contains the best solutions (in terms of quality), and for updating the probabilities given for each potential facility within the construction phase; this procedure is repeated until the maximum number of iterations is reached. The best solution and its corresponding objective function value found for the leader is stored and is presented as the algorithm output.

In the following section, the procedure for locating facilities (constructing leader's solutions) is detailed.

6.2.5.2 Facility Location Procedure

Each iteration starts with the decision of whether to locate a facility. In an iteration t, this decision is conditioned by a probability value $p_i^t, \forall i \in I$. Furthermore, binary variables y_i are taken as independent Bernoulli random variables. At the beginning of the procedure, the initial probabilities are set to $p_i^0 = 0.5, \forall i \in I$, which indicates all the facilities have the same chances to be located. As the procedure goes forward, these probabilities will update favoring the most convenient facilities according to the information provided by the previously constructed samples. For conducting this, the value q_i, the proportion of times that facility is located among the n best solutions, is computed. Finally, for each facility $i \in I$, the probabilities are updated in the following manner: $p_i^{(t+1)} = \lambda(q_i) + (1 - \lambda)p_i^t$, where λ indicates a smoothing parameter.

Every constructed set of located facilities is inspected to assure feasibility; that is, the sum of the capacities associated with the located facilities must be enough to cover the customers' total demand. Hence, once a feasible set of located facilities is obtained, allocation of customers based on their preferences is performed.

6.2.5.3 Customers Allocation Procedure

As mentioned above, three strategies for allocating customers to facilities (solving the follower's problem) were implemented: Optimizer, Construction-p, and Construction-r. In the first, the lower level problem is solved through a commercial software optimizer. Then, in the Construction-p, a greedy function for measuring the candidates based on the customer' preferences is considered. Finally, Construction-r is a variant of the latter. The main difference is that a regret cost is measured for comparing the impact of allocating a customer to its second-most preferred facility instead of the first one.

Optimizer

For this procedure, CPLEX was used to solve the lower level problem for each leader's decision; bilevel feasible solutions resulted under this approach. For the tested instances, the optimizer always reached the optimal solution as long as the leader's solution was feasible in terms of capacity constraints.

Construction-p

To initialize this procedure, the best preference value for each non-allocated customer towards the located facilities is identified. Then, considering these preferences, a restricted candidate list (RCL_1) is created with the facility-customer pairs associated to the $\alpha\%$ best preferences values that can be satisfied by the available capacity. Therefore, one facility-customer pair is randomly selected from RCL_1. In each iteration, the procedure adds to the solution a feasible allocation of a customer to a facility.After each allocation, the available capacity of the facility is updated and the procedure continues until all customers are allocated to a located facility. A pseudocode of this procedure is displayed in Algorithm 6.2.2.

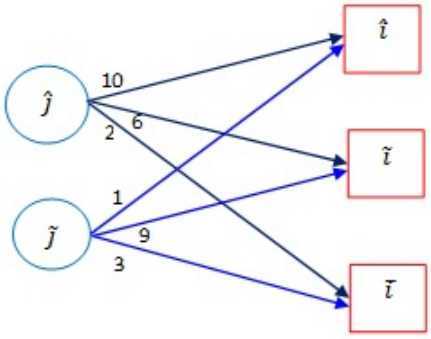
Algorithm 6.2.2 Pseudocode of the *Construction-p* procedure.

procedure CONSTRUCTION-P(α, y)
  $A \leftarrow \{1, 2, ..., m\}$ (set of customers that needs to be allocated)
  $L = \{i \in I : y_i = 1\}$ (set of located facilities)
  $d\_accum_i \leftarrow 0, \forall i \in L$ (accumulative demand for the located facilities)
  $G_j \leftarrow \{g_{ij} : i \in L\}, \forall j \in J$ (set of preferences for the located facilities)
  $x \leftarrow \emptyset$
  while $(A \neq \emptyset)$ **do**
    Randomly selects $j \in A$
    $c^{max} \leftarrow argmax\{G_j\}$
    $c_{min} \leftarrow argmin\{G_j\}$
    $RCL_1 \leftarrow \{i \in L : g_{ij} \geq c^{max} - \alpha(c^{max} - c_{min}), d_j + d\_accum_i \leq b_i\}$
    Randomly selects $i \in RCL_1$
    $x \leftarrow \{(i, j)\} \cup x$
    $A \leftarrow A \setminus \{j\}$
    $d\_accum_i \leftarrow d\_accum_i + d_j$
  end while
  return x
end procedure

Construction-r

As mentioned before, this procedure only differs from Construction-p by the function used for measuring the convenience of including a facility-customer pair in the solution under construction. This function is inspired in a generalized regret measure that intends to simulate the scenario when a customer is allocated to its second-most preferred facility instead of to the first one. For example, consider Figure 6.9, the regret value for the customer $\hat{j}$ would be 4 -since its two greater preferences are 10 and 6, respectively-, that is, $R(\hat{j}) = 4$. On the other hand, for the customer $\tilde{j}$, $R(\tilde{j}) = 6$, which implies it would be more convenient to allocate the customer $\tilde{j}$ to its most preferred facility. If the allocation is not made at this step, there is a risk that in further iterations the most preferred facility does not have the capacity to serve him.

Figure 6.9: Illustrative example of the regret measure.



As described in the illustrative example, the two best preference values for each non-allocated customers to the located facilities are identified. Then, a restricted candidate list (RCL_2) is created using the difference between these two best values for each customer and considering the available capacity in the facilities. Therefore, RCL_2 includes customers with the $\alpha\%$ greatest difference values. From RCL_2, a customer is randomly selected and assigned to its most preferred feasible facility. Available capacity for this facility is updated, and the procedure continues until every customer is allocated to a facility. The corresponding pseudocode of this procedure is depicted in Algorithm 6.2.3.

Algorithm 6.2.3 Pseudocode of the *Construction-r* procedure.

procedure CONSTRUCTION-R(α, y)
  $A \leftarrow \{1, 2, ..., m\}$ (set of customers that needs to be allocated)
  $L = \{i \in I : y_i = 1\}$ (set of located facilities)
  $x \leftarrow \emptyset$
  $d\_accum_i \leftarrow 0, \forall i \in L$ (accumulative demand for the located facilities)
  $G_j^1 \leftarrow \{g_{ij} : i \in L\}, \forall j \in J$ (set of the most preferred preferences)
  $G_j^2 \leftarrow \{g_{ij} : i \in L\}, \forall j \in J$ (set of the second most preferred preferences)
  $R_j \leftarrow \{G_j^1 - G_j^2 : i_1, i_2 \in L\}, \forall j \in J$ (set of the differences between these two best preferences)
  while $(A \neq \emptyset)$ **do**
    $Randomly \quad selects \quad j \in A$
    $c^{max} \leftarrow argmax\{R_j\}$
    $c_{min} \leftarrow argmin\{R_j\}$
    $RCL_2 \leftarrow \{i \in L : g_{ij} \geq c^{max} - \alpha(c^{max} - c_{min}), d_j + d\_accum_i \leq b_i\}$
    Randomly selects $i \in RCL_2$
    $x \leftarrow \{(i, j)\} \cup x$
    $A \leftarrow A \setminus \{j\}$
    $d\_accum_i \leftarrow d\_accum_i + d_j$
  end while
  return x
end procedure

6.2.6 Computational experimentation

In this section, the results obtained from the computational experimentation are discussed. First, the bounds presented in Section 6.2.3 were obtained by an optimizer. Second, the three versions of the proposed CE method were tested. Finally, an analysis from the obtained results and a discussion about the ϵ value introduced in Section 6.2.2 is presented.

6.2.6.1 Computational environment and instance's description

A workstation with an Intel (R) Core processor with 32GB of RAM was used for conducting the computational experimentation. The code was implemented in Visual Studio 2012 in C++ language. The optimizer used for obtaining the bounds and for solving the lower level problem in one of the CE's schemes was CPLEX 12.6.1.

The instances were adapted from the *Capacitated warehouse location* set contained in the well-known Beasley's OR library; these instances include the number of facilities (m) and the number of customers (n). For each customer, its demand and the cost for allocating it to each facility is given. For each facility, the capacity and the fixed location cost are included; the preferences are missing, however. To deal with this issue, we implemented the method provided in [26], which generates random preferences but maintains some rationality with respect to their allocation costs.

We implement an enumerative algorithm for comparing the values given by the reformulation used for obtaining classical lower bounds. The lower level is optimally solved by an optimizer for each upper level's decision. By doing this, bilevel feasible solutions are obtained and the optimal solution is guaranteed.

6.2.6.2 Enumerative algorithm

Due to the limitations of the enumerative algorithm, we decided to test only with some small size instances. For example, for the 16×50 instances, there are only 26,333 feasible solutions; but, for a set of dimensions 25×50 there are 33,554,431 feasible solutions. Moreover, the lower level is optimally solved for each of these solutions. This clearly limits the capability of the enumerative algorithm for solving medium or large size instances. Then, we decided to prove some instances of size 15×30 . Also, the aim of this work is to measure the efficiency of the proposed bounds and for doing that, the optimal solution of the tested instances is required.

Table 6.4: Results obtained from the computational experimentation with homogeneous production capacity.

	Enumerative algorithm		**Classical lower bounds**		
Instance	**Optimum**	**Time (s)**	**Bound**	**Time (s)**	**Gap (%)**
16×50-41	1,518,736	1,697.72	964,895	1.02	36.47
16×50-42	1,640,170	1,072.51	1,016,155	1.11	38.05
16×50-43	1,599,687	907.21	1,064,678	2.20	33.44
16×50-44	1,648,478	1,001.80	1,129,726	7.19	31.47
16×50-51	1,416,440	1,186.43	1,014,038	1.56	28.41
16×50-61	1,307,693	1,230.22	932,616	0.64	28.68
16×50-62	1,305,170	1,288.15	977,799	0.56	25.08
16×50-63	1,322,223	1,234.80	1,010,808	1.02	23.55
16×50-64	1,405,077	1,260.28	1,042,331	2.02	25.82

From Table 6.4, it can be appreciated that -as it is expected- the enumerative algorithm consumes a significant amount of time for solving small size instances. The scheme for obtaining the classical lower bounds is very fast. However, the optimality gaps are very large -between 23.55% and 38%-. Furthermore, when considering instances with different structure, the behavior very similar -see Table 6.5-.

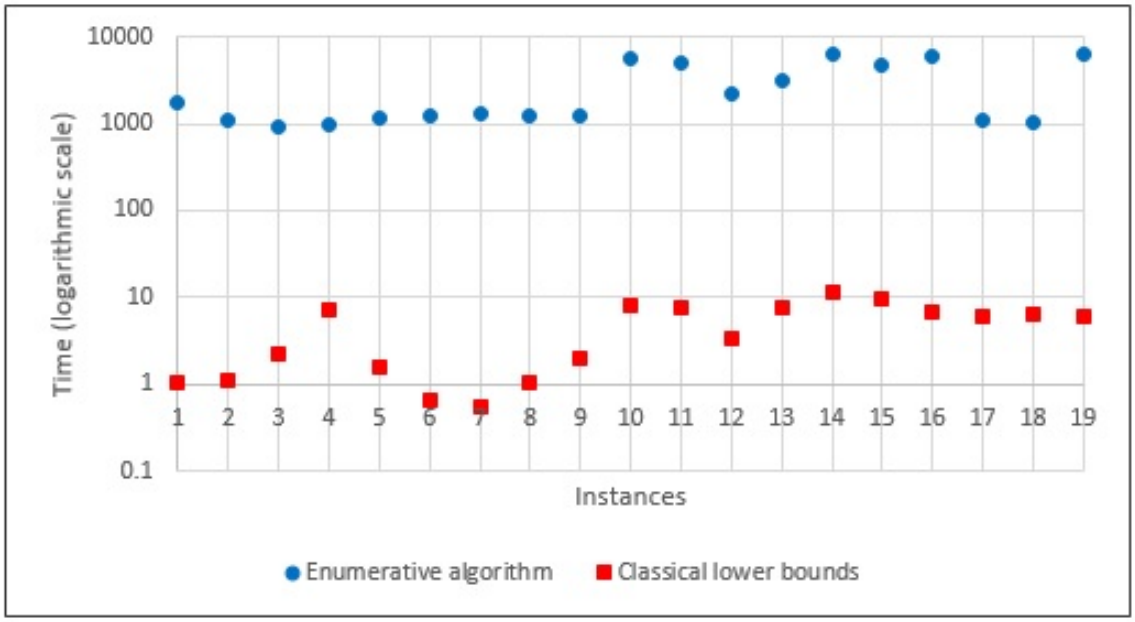
Figure 6.10 in a logarithmic scale- the required time for the two considered schemes is shown. The findings are very intuitive, that is, the enumerative algorithm consumes more time than the other method. The first reformulation, which gives poor quality bounds, is the less time consuming for the first set of

Table 6.5: Results obtained from the computational experimentation with heterogeneous production capacity.

	Enumerative algorithm		Classical lower bounds		
Instance	**Optimum**	**Time (s)**	**Bound**	**Time (s)**	**Gap (%)**
15×30-p7	5,102	5,705.18	4,366	8.14	14.43
15×30-p8	8,908	5,022.95	7,926	7.50	11.02
15×30-p9	3,363	2,235.52	2,480	3.33	26.26
15×30-p10	24,012	3,076.26	23,144	7.63	3.61
15×30-p12	4,417	6,186.57	3,711	11.14	15.98
15×30-p13	4,569	4,837.69	3,760	9.67	17.71
15×30-p14	6,700	6,026.05	5,965	6.92	10.97
15×30-p15	8,909	1,079.64	7,816	5.94	12.27
15×30-p16	12,262	1,024.27	11,543	6.39	5.86
15×30-p17	10,700	6,299.82	9,884	5.97	7.63

instances.

Figure 6.10: Consumed time (in seconds).



Given that the results obtained with the reformulation provides lower bound of poor quality, and that computational effort of the exact procedure is significantly large, we decided not to test the other instances with these two procedures.

Therefore, the instances were tested with the reformulations R1 and R2 of the relaxed BCFLP, and the cross entropy method. the constructed set contains 49 instances classified into four subsets, as summarized in Table 6.6.

Table 6.6: Classification of instances.

Instance label	# Instances	m	n
Set A	13	16	50
Set B	12	25	50
Set C	12	50	50
Set D	12	100	1000

6.2.6.3 Computing a bound

The bound given by the reformulations of the bilevel model based on the relaxation of the follower's variables is computed. Remembering that R1 is given by equations (6.70)-(6.81) and R2 is defined by (6.88)-(6.100), both reformulations reach the same bound. In Tables 6.7 and 6.8, the leader's objective function reached by CPLEX and the respective time consumed (in seconds) by both reformulations are presented when solving Sets A and B. For these tests, a maximum time limit of 3 hours was set as a stopping criterion.

Table 6.7: Bounds obtained from the relaxation of the set A.

Instance	Bound	Time_R1	Time_R2
16x50-41	1512700.00	196.94	322.92
16x50-42	1737580.00	197.77	930.63
16x50-43	1604880.00	86.28	671.41
16x50-44	1675090.00	89.11	1141.61
16x50-51	1437680.00	49.41	1768.69
16x50-61	1289980.00	15.39	55.78
16x50-62	1309990.00	17.16	46.66
16x50-63	1322760.00	25.83	118.42
16x50-64	1385040.00	30.72	32.98
16x50-71	1248140.00	15.22	4.72
16x50-72	1248140.00	5.58	5.39
16x50-73	1248140.00	5.11	3.58
16x50-74	1248140.00	2.47	6.00

Both reformulations are significantly affected by the increase in the size of the instance. Moreover, R1 and R2 were incapable of obtaining bounds for sets C and D due to memory limitations. Therefore, this scheme for obtaining bounds can only be used on limited size instances. Later, we will compare the quality of the bounds against bilevel feasible solutions. For instances 16×50-71 to 16×50-74 (see 6.7), the obtained bound is the same, but this is due to the particular instance' structure; the only difference among them is that fixed costs and customer preferences are modified -this is the reason that leads to the same value. This structure remains in instances 25×50-101 to 25×50-104 (see 6.8).

6.2.6.4 The results obtained by the Cross Entropy method

The results obtained from the three different versions of the CE method proposed in Section 6.2.5 are presented. The use of CPLEX for solving the lower level problem yields bilevel feasible solutions,

Table 6.8: Bounds obtained from the relaxation of the set B.

Instance	Bound	Time_R1	Time_R2
25x50-81	1418770.00	10800.00	10800.00
25x50-82	1419720.00	10800.00	10800.00
25x50-83	1416500.00	10800.00	10800.00
25x50-84	1483020.00	10800.00	10800.00
25x50-91	1291310.00	8592.06	10800.00
25x50-92	1319430.00	4815.75	10800.00
25x50-93	1329850.00	976.80	10800.00
25x50-94	1312320.00	816.98	10800.00
25x50-101	1248140.00	567.25	55.48
25x50-102	1248140.00	615.67	33.38
25x50-103	1248140.00	432.86	64.42
25x50-104	1248140.00	98.53	25.31

while the use of the construction methods detailed in subsection 6.2.5.3 provides bilevel attainable solutions (introduced in Section 6.2.2).

Before conducting the experimentation, the parameters' calibration was done. The parameters involved in the CE are as follows:
1. the size of the set of solutions (N);
2. the number of elite solutions selected from the set of N solutions (d);
3. the smoothing parameter in the CE (λ);
4. the allowed degree of randomness for creating the RCL in the construction methods (α); and
5. the maximum number of iterations (*maxIter*).

For calibration purposes, we used CALIBRA, a procedure created for tuning parameters in heuristic-based algorithms. This automated procedure combines Taguchi's design of experiments and a local search procedure. For more detailed information, see [2]. Hence, as input to that procedure we selected some values or ranges for varying the parameters. For N, three values were considered: n, $m + n$ and mn. For d, we selected 20% and 50% of the size of N. The λ used in the CE framework and the α considered in the construction methods were varied from the interval $[0, 1]$ with increments of 0.1. Finally, three options for $maxIter$ were taken -50, 75 and 100. The parameter setting based on the CALIBRA results are shown in Table 6.9.

Table 6.9: Parameter setting.

Instance label	N	d	λ	α	$maxIter$
Set A	$m + n$	20%	0.8	0.9	75
Set B	$m + n$	20%	0.8	0.9	75
Set C	$m + n$	20%	0.2	0.3	100
Set D	n	20%	0.2	0.3	100

Once the parameters were calibrated for each set of instances, the computational experimentation was

conducted. Due to the randomness considered in the proposed algorithm, 10 replicates for each instance were run. The average of the results are displayed in Tables 6.10-6.13, in which three main blocks can be observed, each of them corresponding to the schemes proposed for solving the lower level problem. The block for the results obtained by the Optimizer scheme indicates the following: the "Best" ("Worst") column contains the best (worst) leader's objective function value, the "Average" column displays the mean value from the final set of solutions obtained by the CE method, and the "Time" column shows the required time (in seconds) for solving the instance. Then, for the other two blocks, Construction-p and Construction-r, the average deviations for the Best and Average values are shown. The formula used for computing the average deviation is the following: $value = 100 * \frac{((Construction - Optimizer))}{Optimizer}$. Furthermore, the required time is shown for each scheme.

Table 6.10: Results obtained by the CE method over the set A.

	Optimizer				Construction-p				Construction-r			
Instance	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**
16×50-41	1518740	1518740	1544660	191.26	0.66	0.85	0.29	2.22	-0.90	-0.77	2.81	2.53
16×50-42	1643020	1643020	1697990	217.63	3.22	3.31	5.06	2.10	-3.25	-1.09	-2.14	2.53
16×50-43	1599690	1599690	1701190	183.52	4.66	8.69	9.78	2.14	-7.65	-6.45	-1.85	2.62
16×50-44	1640690	1640690	1690210	223.63	2.72	3.24	0.74	2.19	-0.76	-0.04	-1.28	2.49
16×50-51	1416440	1416440	1467310	116.34	-0.22	-0.22	-1.44	1.71	0.71	0.80	0.69	2.02
16×50-61	1285510	1285510	1307930	92.81	-0.02	-0.02	-0.08	1.57	-0.52	-0.52	-0.16	1.98
16×50-62	1316000	1316000	1328290	93.24	-1.01	-1.01	0.80	1.55	-0.23	-0.23	-0.28	1.89
16×50-63	1322760	1322760	1391930	90.22	0.00	0.00	-0.92	1.61	0.00	0.00	-1.73	2.00
16×50-64	1379490	1379490	1393190	91.31	1.09	1.09	5.33	1.55	0.88	0.88	0.39	1.92
16×50-71	1310310	1327679	1358650	23.84	-4.74	-0.65	-1.59	1.64	0.00	0.00	0.74	2.06
16×50-72	1248140	1248140	1387340	23.75	0.00	0.00	0.00	1.52	3.71	3.71	0.00	1.85
16×50-73	1248140	1248140	1338330	23.88	0.00	0.63	0.00	1.54	6.28	6.28	0.00	1.90
16×50-74	1248140	1248140	1309700	23.51	0.00	0.00	-4.70	1.46	4.93	4.93	3.32	1.75

Table 6.11: Results obtained by the CE method over the set B.

	Optimizer				Construction-p				Construction-r			
Instance	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**
25×50-81	1285780	1332652	1420690	187.07	8.25	7.53	9.64	3.20	3.50	0.94	1.20	3.77
25×50-82	1317690	1321149	1464700	200.49	1.41	1.15	-2.53	3.11	2.79	4.58	-1.08	3.70
25×50-83	1386660	1404591	1460210	172.15	-0.43	-1.00	-1.70	3.12	-0.03	-0.32	2.29	3.70
25×50-84	1296910	1383558	1533670	183.38	3.13	6.68	1.28	3.15	-0.95	-1.70	0.79	3.72
25×50-91	1245750	1334335	1406920	105.23	-1.10	-2.98	0.97	2.77	-0.02	-0.19	0.14	3.43
25×50-92	1250750	1260838	1297780	115.01	0.97	0.18	0.09	2.73	0.02	0.25	-0.32	3.25
25×50-93	1251590	1267030	1386590	121.78	2.62	7.29	9.72	2.76	0.52	2.09	8.43	3.22
25×50-94	1235870	1237775	1347250	123.47	8.73	8.57	3.15	2.61	1.35	1.23	-0.70	3.10
25×50-101	1248140	1277096	1436970	33.58	0.00	0.23	-3.37	2.60	4.48	2.28	-3.37	3.01
25×50-102	1248140	1292498	1515950	33.61	0.00	2.37	-5.81	2.58	4.35	2.24	-9.72	3.14
25×50-103	1284260	1288151	1375480	33.60	0.00	-0.06	-3.45	2.70	0.00	1.15	0.50	3.17
25×50-104	1248140	1248140	1500280	33.61	0.00	0.00	1.22	2.48	4.85	4.85	-10.00	2.96

Tables 6.10-6.13 show positive and negative values appear in the Construction-p and Construction-r blocks; bilevel attainable solutions do not guarantee to be bilevel feasible. On the contrary, solutions obtained in the Optimizer block are bilevel feasible.

Some findings are stated next. For the set A (see Table 6.10), the average time for reaching bilevel feasible solutions is 107.3 seconds per instance. Regarding the bilevel attainable solutions within the Construction-p framework, the average deviations vary from -4.74 to 4.6, taking 1.75 seconds. This procedure found bilevel feasible solutions in four out of thirteen instances. On the other hand, the Construction-r scheme shows a similar performance with a range from -7.65 to 6.28 requiring 2.12 seconds for solving the problem and obtaining feasible ones in two out of thirteen instances. Furthermore, the worst values in Construction-r vary from -4.70 to 9.78, and in Construction-p between -2.14

Table 6.12: Results obtained by the CE method over the set C.

	Optimizer				Construction-p				Construction-r			
Instance	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**
50×50-111	1622350	1653775	1746090	162.95	0.45	0.18	0.47	8.70	-0.93	-0.45	-0.07	10.46
50×50-112	1508580	1548539	1694920	170.67	2.18	3.89	1.17	8.66	1.05	1.85	-1.49	10.26
50×50-113	1785740	1858121	1975270	164.12	-2.44	-3.15	-2.15	8.52	-3.69	-1.49	2.02	10.37
50×50-114	1730630	1903997	2111150	168.11	6.09	1.58	3.55	8.49	0.05	-2.79	-0.53	10.11
50×50-121	1527760	1666469	1823710	153.51	4.19	0.58	0.68	8.37	9.42	1.31	-1.18	10.16
50×50-122	1582400	1787738	2010260	144.00	1.13	1.77	-0.13	8.30	7.05	1.43	-0.79	10.05
50×50-123	1634770	1677889	1794690	144.32	1.14	-0.18	0.59	8.36	-1.44	-1.03	-2.58	10.21
50×50-124	1748210	1839056	2053000	149.08	1.27	-0.01	-2.23	8.27	-1.21	-0.41	-1.93	10.10
50×50-131	1504200	1547155	1676960	55.88	-2.14	-0.77	-0.27	8.32	-0.65	-1.79	-0.25	10.13
50×50-132	1711810	1764714	1849340	56.02	-5.24	-1.79	0.56	8.31	-3.39	-2.99	0.15	10.04
50×50-133	1574220	1660547	1784690	55.98	3.00	2.28	1.68	8.22	4.42	1.93	1.43	9.98
50×50-134	1600350	1720798	1882520	55.92	8.14	3.89	0.66	8.24	7.98	2.56	0.75	10.11

Table 6.13: Results obtained by the CE method over the set D.

	Optimizer				Construction-p				Construction-r			
Instance	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**	**Best**	**Average**	**Worst**	**Time**
100×1000-a1	80822500	89091800	99840400	1055.65	-1.79	-4.63	-6.99	77.39	-10.40	-4.69	-0.95	103.38
100×1000-a2	75313600	88666890	97543600	1064.32	1.82	-1.03	-1.13	80.26	1.32	-2.38	-1.46	103.25
100×1000-a3	69542400	92042730	98428800	1035.64	7.08	-1.72	-2.16	77.62	9.71	-3.55	1.02	103.48
100×1000-a4	79891300	87541290	98104200	1014.37	-0.25	-1.10	-0.64	80.87	3.48	5.24	0.26	105.33
100×1000-b1	55453300	57340260	59605400	1056.32	-2.11	0.91	1.80	80.93	-0.97	0.14	-1.86	106.26
100×1000-b2	53409500	57153520	61518400	1011.68	-2.98	-0.36	-5.21	79.80	2.31	0.98	-3.06	107.30
100×1000-b3	54476400	56782440	58724700	1019.56	-1.40	-1.28	1.53	82.35	1.89	0.36	-0.03	106.87
100×1000-b4	55340300	57401730	58940000	1028.82	-0.31	0.19	2.98	81.85	-2.43	-0.48	0.86	106.82
100×1000-c1	47124900	48564430	49939500	1037.25	0.16	0.02	1.89	80.41	-0.82	0.38	0.05	106.81
100×1000-c2	47383200	49082320	50170300	1036.41	2.24	-1.13	-0.16	81.81	-1.77	-1.15	-0.69	107.33
100×1000-c3	45506800	49002530	51167800	1027.08	1.52	-0.40	-0.82	82.11	3.94	-0.09	-0.20	107.67
100×1000-c4	48343700	49439660	50862700	1014.64	-0.38	-0.01	-0.65	79.83	-4.05	-0.85	-0.03	107.39

and 3.32.

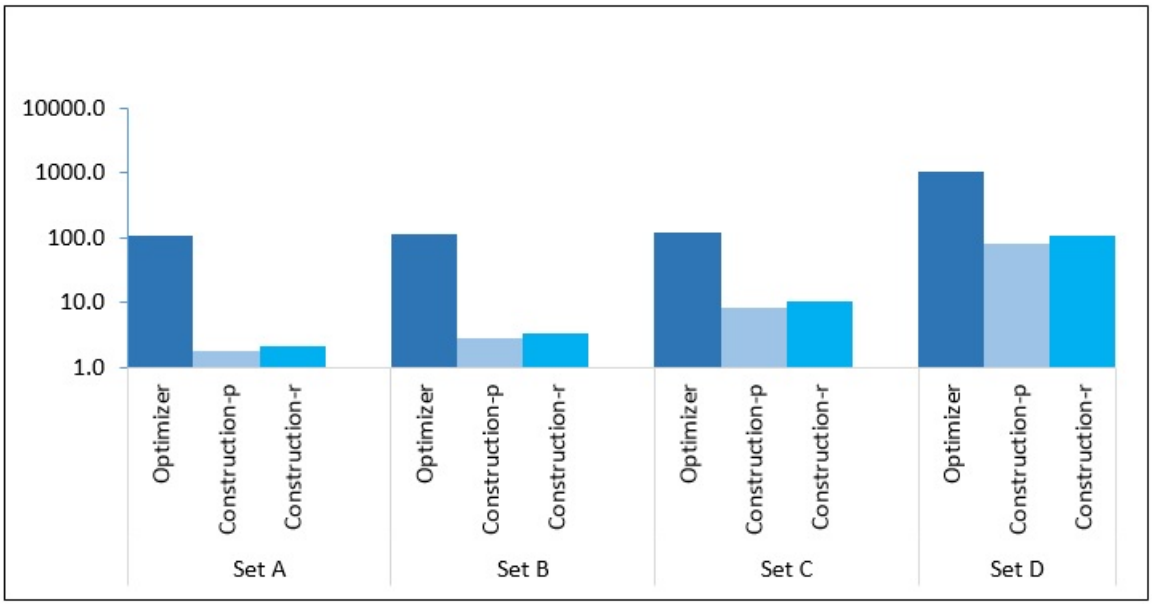
From Tables 6.11 and 6.12, it appears the time was slightly increased (as expected). It took 111.92 and 123.38 seconds on average for solving the sets B and C under the Optimizer scheme, respectively. Both constructions procedures demand similar time. The Construction-p procedure reaches the best feasible solution, however in four out of twelve instances for set B. In addition, the higher values of the average deviations are not greater than 10.00 and the worst value varies between -10.00 and 9.64. For the set D (Table 6.13), the behavior of the CE method is maintained despite a significant increase in the number of customers. The quality of the bilevel attainable solutions is not compromised, and the required time is still reasonable. On the contrary, obtaining bilevel feasible solutions seems to be a demanding task. This fact confirms an accurate alternative manner for solving the problem is useful.

In Figure 6.11, the required average time for solving an instance by each scheme is shown. A logarithmic scale was used for improving the readability. Both approaches employed for obtaining bilevel attainable solutions significantly reduce the computational time needed for obtaining bilevel feasible ones. In the next subsection, an analysis of the efficiency of bilevel attainable solutions is presented.

6.2.6.5 Analyzing the efficiency of the bilevel attainable solutions

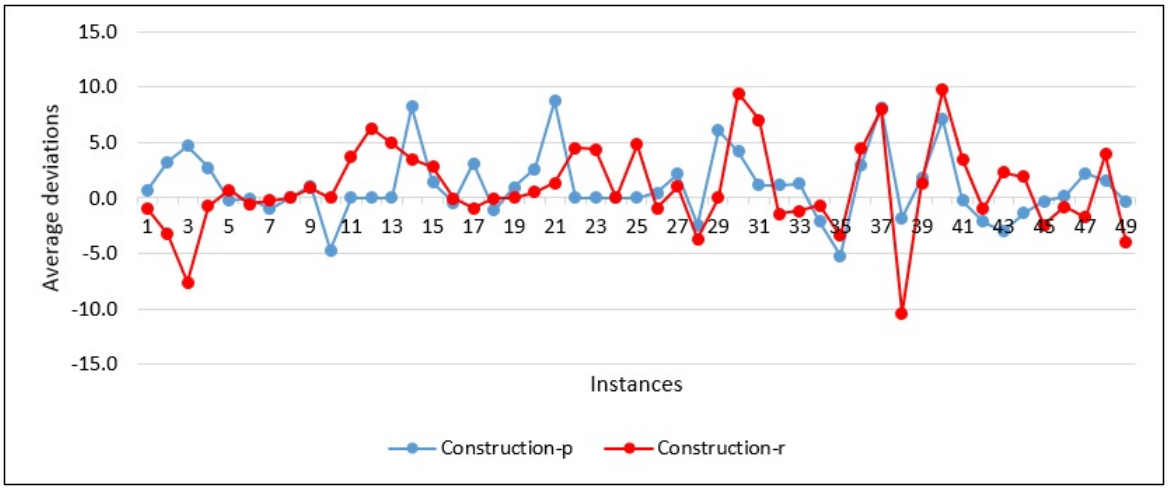
In Section 6.2.2, bilevel attainable solutions were defined as solutions in which the lower level problem has not been solved in an optimal way. Good lower level solutions are desired, however, to have an efficient set of bounds. Hence, a small value of ϵ introduced in Definition 3 is needed for approximating accurately the inducible region; it represents the allowed level for not reaching the optimal value.

Figure 6.11: Computational time required for the three schemes of the CE method.



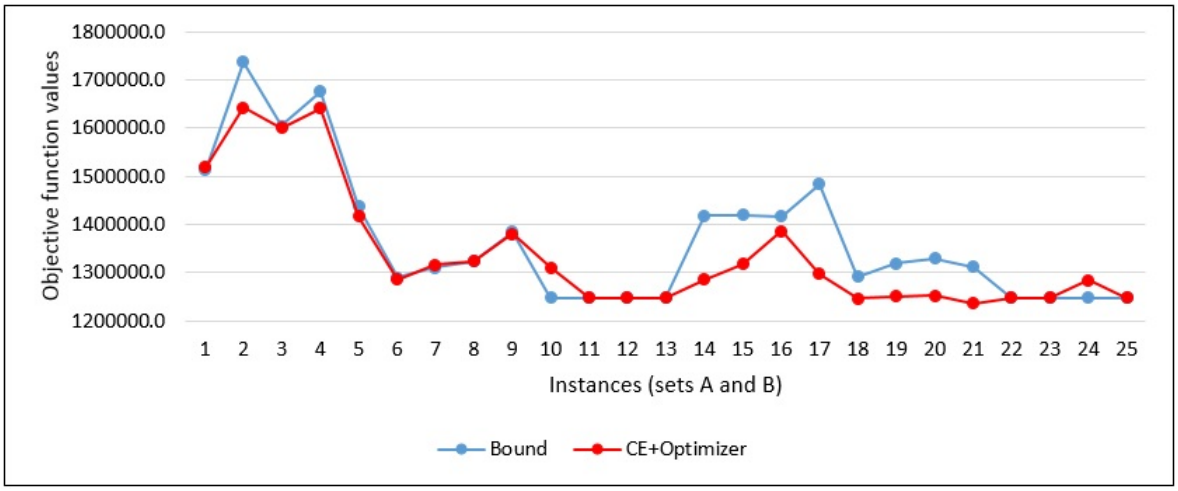
In Figure 6.12, for 49 instances, the average deviations of the bilevel attainable solutions with respect to the bilevel feasible ones are plotted. Suppose that 0.05 is the value set for ϵ. Therefore, for the Construction-p, in 43 out of 49 instances the attainable solutions are within the allowed range. For the Construction-r, a similar performance is reported with 42 out of 49 instances in which the attainable solutions are accepted. The latter observations confirm the construction procedures proposed as alternative methods for dealing with the lower level problem are very good options when bilevel feasible solutions cannot be obtained.

Figure 6.12: Behavior of the attainable solutions against feasible ones.



Finally, to conclude the analysis of the computational results, the quality of the bounds obtained by R1 and R2 are compared with the best bilevel feasible solutions reached by the CE under the Optimizer approach. The leader's objective function values for sets A and B are plotted in Figure 6.13. The results support the discussion presented in Section 6.2.4, in which the identification of lower or upper bounds could not be stated for this problem. In Figure 6.13, it appears that sometimes the value for the bound is higher than the value corresponding to the bilevel feasible solutions and in other occasions it is lower.

Figure 6.13: Comparing the bounds.



Chapter 7

Conclusions and Further Research Directions

In this doctoral thesis we studied several bilevel facility location problems with customer preferences. First, we analyzed the case without capacity restrictions. A hybrid evolutionary algorithm (EA) with path relinking (PR) for solving the uncapacitated facility location problem (UFLBP) with customer preferences is proposed. A computational analysis which involves three approaches for dealing with preference orderings in the lower level problem, is presented. First the EA is implemented in the same manner as it was proposed in [22]. Then, the hybrid EA+PR is developed for exploiting the bilevel structure of the problem. Finally, a variant of the hybrid named EA+PRw was tested. In the variant considered, the lower level problem is not solved at each iteration of path relinking. According to computational experimentation, it seems that the computational time required to solve the lower level is very short. For example, as we shown in section 3.1.3, EA+PR has a better performance than EA+PRw; and regarding with the computational time, there is an insignificant difference between both algorithms. However, if solving the lower level requires an excessive computational time, then it will be convenient not to solve the lower level for each leader's solution. Thus, semi-feasible solutions will be recommended for guiding the search. As a result of the latter approach, it is observed that the quality of the solutions would not improve significantly, in spite of that the computational time will decrease noticeably. Hence, the importance of having alternative options for optimally solving the lower level problem instead of using commercial optimization software is remarked. Nevertheless, this issue depends on the structure of the problem being studied.

Future research directions could involve the advantage of some particular properties existing in the lower level that may help to sort this problem. That is to avoid a blind search in the lower level, because of the use of semi-feasible solutions due to the fact that the lower level is not solved in several consecutive iterations in the methodology selected. For example, recently in [127] an attempt for approximating the lower level optimal solutions without explicitly solving it is presented. The approximation consists in create quadratic functions based on a set of initial bilevel feasible solutions. Then, for an upper level solution, the lower level reaction is approximated by using those functions. After a fixed number of steps following this approach, the lower level is optimally solved for each upper level solution. By doing this, bilevel feasible solutions are obtained replacing the bilevel semi-feasible ones. These ideas can be used in the EA+PRw for improving its performance.

Another option is to apply a decomposition approach based on the primal and dual relationships of the problem as proposed in [94], that describes how it is applied for obtaining high quality solutions in a

short term generation planning. In that problem, the computational effort is increased in an exponential manner as the number of the components involved in the problem also augments. Hence, the decomposition approach shown that it is a good alternative for dealing with high complexity problems.

Also, we explored a variant of the p-median problem. This variant is based on the assumption that customers are free to choose the located facility that will serve them (p-median BPO). In this problem, two different decision makers consider a predefined hierarchy. The first decision maker (leader) chooses which facilities are to be located, and the second one (follower) controls allocation of customers to the located facilities. The latter decision is based on the assumption that the customers have an ordered list of preferences associated with all possible facilities and they try to optimize those preferences. Then, the leader locates the facilities that minimize distribution costs.

To solve the problem, two single-level reformulations were presented. The first one follows a well-known technique for reformulating bilevel problems based on the primal-dual relationships of the follower's problem. The second one is based on closest assignment constraints that express consideration of the preferences. Also, an adaptation of a single-level reformulation in the literature for the SPLPO (which a similar problem to the p-median BPO) is made. Numerical results show that the use of optimization software for solving any of the reformulations is inefficient for medium and large-size instances. Furthermore, a hybrid heuristic procedure that combines scatter search and GRASP for leader's solutions considering the optimal response of the follower (Stackelberg's equilibrium) was developed. The algorithm's efficiency was validated using the same set of instances used for the reformulations and via a comparison against other heuristic algorithms (a scatter search with random construction, a scatter search with greedy construction, and a genetic algorithm). The obtained results indicate that the performance of the proposed heuristic is very good in terms of the value of the leader's objective function and the required time. The algorithm reaches the optimal solution or the best known values within a short computational time and obtains low average percent deviation values for all instances. Moreover, it can be seen from the reported results that the hybrid heuristic's performance does not depend on the instance size. It is well known that the quality of the solution is directly related with the time that the algorithm is executed (see [87]), but the hybrid heuristic reported very good results in both criteria.

Further research direction may be to take advantage of the bilevel structure of the problem and consider an optimality condition decomposition approach, as the one proposed in [63]. The latter approach is suitable when a decomposable form could be find in a specific problem involving complexity constraints. In this case, we have an optimization problem within the constraints under a decomposable structure. Other decomposition approaches have been applied for solving bilevel problems (see, [141], [121], [110] and [53]).

Furthermore, we proposed for the first time in the literature a maximal covering problem that considers customer preferences (MCLP). The problem considers two decision levels, one associated with facilities location, and the other related to allocation of customers to opened facilities. The upper level maximizes the demand covered, while the lower level maximizes customer preferences. We propose a genetic algorithm (GA) to find good quality lower bounds for the problem. The algorithm is tested with a set of randomly generated instances. To evaluate the proposed algorithm, we present two single-level reformulations of the bilevel problem. Both reformulations are based on the strong duality property of the follower's problem. One of the reformulations is based on equalizing the primal and the dual objective functions of the follower's problem, and the other introduces in the problem the complementary slackness conditions. Since the upper bounds of the second reformulation are tighter than the

upper bounds of the first one, it is used to evaluate the quality of the solutions obtained by the GA. The reformulation was executed using the commercial software CPLEX with a three-hour time limit.

The performance of the proposed GA depends largely in a proper selection of the algorithm parameters. The algorithm requires to determine the values for three parameters: population size, number of generations, and probability of selecting the crossover operator. To set the best possible values, a calibration of these parameters was conducted. This calibration was done through a full factorial design with predefined values for each parameter. Tests results showed that populations of 100 and 200 chromosomes were adequate to obtain good quality lower bounds. Finally, comparing quality and execution times, the population size was set to 100 chromosomes. Then, an analysis of the dominance among combinations of crossover probabilities and number of generations was done. After this analysis, the best configuration of the parameters was selected for the computational experiment.

The proposed GA was tested with six sets of instances of different sizes. The GA always obtained lower bounds of very good quality for all sets. The average gap of the GA solutions when compare to the optimal or best known solution was 0.085%, for medium-size instances, and 0.253% for large-size instances. The proposed GA, obtained better lower bounds than CPLEX for five instances. With respect to CPU time, the proposed GA required much less CPU time than CPLEX. It is important to mention that CPLEX was not able to find the optimal solution for any of the largest-size instances in a three hour time limit. In general, we observed that the proposed GA is robust, since the worst average gap never exceeded 1.77%.

To reduce CPU time and improve solutions quality obtained with the two reformulations and the GA mentioned above, a single-level equivalent model and a hybrid heuristic were presented. The equivalence between these two models was also discussed. Two heuristic algorithms were proposed to obtain lower bounds to the optimal solution of the problem: a GRASP heuristic and a hybrid GRASP-Tabu heuristic that replaces the local search phase of the GRASP heuristic with a Tabu search procedure.

The single-level reformulation was solved for all instances using mathematical programming software (FICO XPRESS) with two purposes: (1) to evaluate the quality of the solutions obtained by the proposed heuristics and (2) to evaluate the efficiency of an exact method as instance size increases.

The heuristic solutions were compared with solutions of the single-level reformulation of the problem. According to the results of the computational tests, the two proposed heuristics provided good quality solutions. In 51 out of the 60 test instances, the proposed heuristics found the optimal solution in at least one of the five executions. In addition, for the remaining nine instances, for which the optimality of the solutions could not be verified within the three hour time limit using FICO XPRESS, the heuristics provided better lower bounds than those obtained with FICO XPRESS. However, the proposed GRASP-Tabu hybrid heuristic outperformed the GRASP heuristic because it is more robust without significantly increasing CPU time. The results indicate that the enumerative effort required by FICO XPRESS increases considerably as instance size grows. In preliminary tests, for the largest instances, FICO XPRESS required more than 15 hours. Note that, in the worst case, the time required by the proposed heuristic never exceeds 250 seconds, thereby demonstrating the efficiency of the proposed heuristics.

Metaheuristic methods and cutting-plane methods are very useful to design special purpose exact methods. Metaheuristics can provide high-quality incumbents and primal bounds in Branch & Bound methods, and cutting-plane methods improve dual bounds. Together, they might reduce the enumerative

effort of an exact method. A future direction of this research can be the combination of heuristics and cutting planes for the design of an exact method for the maximal covering location problem with customer preference ordering. Another future research direction can be to consider heuristic methods for the bilevel formulation of the problem based on a two-player, two-stage strategic game with perfect information, similar to the ones proposed in [118].

Another extension of the problem that we investigated is the capacitated version. First, we studied the problem when demand is unitary (CFLBP). Then, we exploited the properties of the lower level to solve it to optimality, this is a transportation problem. We developed a reformulation of the problem and proposed a genetic algorithm where of the lower level problem is solved with simplex method of the transportation problem. Numerical results showed that the genetic algorithm had a good performance.

As short-term future work, we will perform an experiment design in which the probability of crossing and/or mutation is modified. Also, we analyzed the data and everything seems to indicate that there is no effect caused by the seeds. As the statistical analysis can be more complex due to lack of independence if the initial population is fixed, then the initial population will be created each time in the final experimentation. On the other hand, we intuit that preferences influence calculation times and target value. So we have thought that two groups of problems can be considered. The first group would be the current problems, in which the preferences are different for each problem because they are generated individually. The second group, to respect the idea of Holmberg, we will choose a set of preferences in each block of problems with equal costs.

Furthermore, the behavior of the second level decision makers can be different depending on the relationship between the customers. When the second level objective function is defined as the sum of the preferences we assume that there exists cooperation between customers. They accept a solution which is "good" for all of them because minimize the sum of preferences. When a facility with capacity to attend three customers is preferred by four customers, it is not clear that the rejected customer will accept that for the common good. One possibility is to generalize the follower's problem and to consider a ordered p-median problem with λ weights which represent different criteria to the compute the optimal allocation $\{x_{ij}\}$. Another possibility is to include constraints that determine a unique allocation at the second level. For instance, a leader's decision on facility location $\{y_i\}$ would be feasible if and only if all of the customers are attended by their preferred open facility. In this case, it is interesting to prove which assumptions on the matrix $G = (g_{ij})$ guarantee that there exists a feasible solution.

Later, we studied the problem with generalized demand (BCFLP). This version of the problem has not been investigated. The difficulty that this problem presented is because the problem of the lower level corresponds to the generalized assignment problem that is NP-hard. Due to the high complexity that exists when dealing with the lower level problem, which is NP-hard, alternative concepts were introduced for proposing efficient solution methods. We introduced bilevel attainable solutions as good options when obtaining bilevel feasible solutions are very difficult or impossible.

The obtained results from the computational experimentation show that the classical reduction of bilevel programming problems gives poor quality lower bounds. It is well-known that this behavior takes place because the objective function of the lower level is not being taken into consideration during the decision process. In order to compare the obtained bounds, an enumerative algorithm was implemented. Also, a bound was developed from relaxing the integrity conditions in the lower level problem. The obtained bound cannot be stated as an upper or lower bound, however; a discussion is

presented in Section 6.2.4.

A Cross Entropy method is proposed for solving the problem under study. Three different approaches for solving the lower level were considered; the first one obtains bilevel feasible solutions, while the other two reach bilevel attainable solutions. The good performance of the proposed algorithm and the suitability of the bilevel attainable solutions is validated from the computational experimentation. For instance, in [6] a heuristic algorithm is applied to solve a lower level transportation problem. Therefore, a comparison with bilevel feasible solutions is made, and the unpredictable behavior of the bilevel problem is shown.

Numerical results show that if a small tolerance value of 5% is considered, then the CE method found acceptable solutions more than 86% of the time. Moreover, the required time was significantly reduced. Hence, bilevel attainable solutions are good alternatives for handling bilevel problems with a complex lower level problem.

Two straightforward further research directions are identified: (1) analyze techniques that allow determination of the ϵ value during the lower level's resolution; and (2) employ state-of-the-art methods for solving the lower level problem in an optimal manner to find bilevel feasible solutions for the larger-size instances. The first is to consider inelastic demands -that is, if a customer is not allocated to his most preferred facility, his demand will be reduced, affecting the leader's objective function. In this case, the follower's variables denote the fraction of customer's demand satisfied. Another idea is to add constraints that balance demand among the located facilities. Finally, we will consider a user's equilibrium in the lower level for simulating the case when all the customers are independent from each other, but have a constraint that links them all together. The latter can be seen as multiple non-independent followers.

Appendix A

Appendix

Published papers

1. **Casas-Ramírez, M.S.**, Camacho-Vallejo, J.F. Solving the p-median bilevel problem with order through a hybrid heuristic. *Applied Soft Computing* 60 (2017) 73-86. doi.org/10.1016/j.asoc.2017.06.026 (Indexed by SCI IF 3.541).

2. Díaz, J.A, Luna, D.E., Camacho-Vallejo, J.F., **Casas-Ramírez, M.S.**. GRASP and hybrid GRASP-Tabu heuristics to solve a maximal covering location problem with customer preference ordering. *Expert Systems With Applications* 82 (2017) 67-76. doi.org/10.1016/j.eswa.2017.04.002 (Indexed by SCI IF 3.928).

3. **Casas-Ramírez, M.S.**, Camacho-Vallejo, J.F.,Martínez-Salazar, I.A. Approximating solutions to a bilevel capacitated facility location problem with customer's patronization towards a list of preferences. *Applied Mathematics and Computation* 319 (2017), 369-386. doi.org/10.1016/j.amc.2017.03.051 (Indexed by SCI IF 1.738).

4. Maldonado-Pinto, S., **Casas-Ramírez, M.S.**, Camacho-Vallejo, J.F. Analyzing the Performance of a Hybrid Heuristic for Solving a Bilevel Location Problem under Different Approaches to Tackle the Lower Level. *Mathematical Problems in Engineering*, Vol. 2016, ID 9109824, 10 pages. doi.org/ 10.1155 /2016/9109824 (Indexed by SCI/JCR IF 0.644).

5. **Casas-Ramírez, M.S.**, Camacho-Vallejo, J.F. Analyzing valid bounds for a facility location bilevel problem with capacities. Accepted for publication in *International Journal of Combinatorial Optimization Problems and Informatics*. January 2017. (Indexed by CONACYT).

Submitted papers

1. **Casas-Ramírez, M.S.**, Camacho-Vallejo, J.F., Díaz, J.A, Luna, D.E. The Bi-level Maximal Covering Location Problem. Submitted on *Operational Research: An International Journal* (Second revision). August 2016.

Working papers

1. Calvete, H.I., Galé, C., Camacho-Vallejo, J.F., **Casas-Ramírez, M.S.**. A capacitated facility location problem with customer preferences.

2. Díaz, J.A, Luna, D.E., Camacho-Vallejo, J.F., **Casas-Ramírez, M.S.**. Proveedores logísticos: selección de socios en la formación de redes para 4PL's.

Book chapters

1. Camacho-Vallejo, J.F., **Casas-Ramírez, M.S.**, Miranda, P. The p-Median Bilevel Problem under Preferences of the Customers. In R. Z. Ros-Mercado, et al. (Eds.): *Recent Advances in Theory, Methods, and Practice of Operations Research*, pp. 121-127, UANL - Casa Universitaria del Libro, Monterrey, México, Octubre 2014. ISBN: 978-607-27-0357-5.

2. **Casas-Ramírez, M.S.**, Camacho-Vallejo, J.F. Considerando preferencias de los clientes en problemas de localización de instalaciones. In De La Hoz Reyes, et al. (Eds.): *Avances en Investigación de Operaciones y Ciencias Administrativas*, pp. 29-54, Universidad Simón Bolíval, Barranquilla y Cúcuta, Colombia, 2017.

Research Visits

1. *Universidad de las Américas Puebla, Cholula, Puebla, Mexico*.
8-11 December 2014.
Dr. Juan A. Díaz García and Dr. Dolores E. Luna Reyes.

2. *Universidad de Zaragoza, Zaragoza, Spain*.
October 2015 to February 2016.
Dr. Herminia I. Calvete and Dr. Carmen Galé.

3. *Universidad de las Américas Puebla, Cholula, Puebla, Mexico.*
November 2016 to March 2017.
Dr. Juan A. Díaz García and Dr. Dolores E. Luna Reyes.

Conferences

1. *The p-median bilevel problem under preferences of the customers.* XVII Latin-Iberoamerican Conference on Operations Research / III National Congress of the Mexican Society of Operations Research (6-10 October 2014), Nuevo Leon, Mexico.

2. *The Bilevel Capacitated Facility Location Problem with customer's patronization towards a list of preferences.* V Forum of Disclosure and Technology, Autonomous University of Nuevo Leon (January 2015), Nuevo leon, Mexico.

3. *Problema binivel de localización de plantas capacitadas con preferencias de los clientes.* XIX Latin-American Summer School on Operations Research (23-27 February 2015), Quito, Ecuador.

4. *Cotas para el problema binivel de localización de instalaciones capacitadas con orden.* XXV National School of Optimization and Numerical Analysis (6-11 September 2015), Mexico City, Mexico.

5. *El problema binivel de máxima cobertura.* IV National Congress of the Mexican Society of Operations Research (7-9 October 2015), Chihuahua, Mexico.

6. *El problema binivel de localización de máxima cobertura.* University of Zaragoza (February 2016), Zaragoza, Spain.

7. *The maximal covering location bi-level problem.* I International Workshop on Bi-level Programming (7-11 March 2016), Nuevo Leon, Mexico.

8. *El problema de localización de instalaciones capacitado con preferencias de los clientes.* XVIII Latin-Iberoamerican Conference on Operations Research (2-6 October 2016), Santiago, Chile.

9. *Un algoritmo genético para un problema de localización de instalaciones capacitadas con ordenamiento de los clientes.* V National Congress of the Mexican Society of Operations Research (26-28 October 2016), Tamaulipas, Mexico.

Distinctions

1. Volunteer for the Organizing Committee helping in diverse tasks during the organization of the XVII Latin-Iberoamerican Conference on Operations Research. October 6-10, 2014. Nuevo Leon, Mexico.

2. Full scholarship. XIX Latin-American Summer School on Operations Research, February 23-27, 2015. Quito, Ecuador.

3. Full scholarship. V Winter School in Network Optimization, January 11-15, 2016. Estoril, Portugal.

4. Volunteer for the Organizing Committee helping in diverse tasks during the organization of the I International Workshop on Bi-level Programming. March 7-11, 2016. Nuevo Leon, Mexico.

Bibliography

[1] ABDINNOUR-HELM, S., AND HADLEY, S. Tabu search based heuristic for multi-foor facitily layout. *International Journal of Production Research 38*, 2 (2000), 365–383.

[2] ADENSO-DIAZ, B., AND LAGUNA, M. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research 54*, 1 (2006), 99114.

[3] ALEKSEEVA, E., AND KOCHETOV, T. Genetic local search for the p-median problem with client's preferences. *Diskret. Anal. Issled. Oper. 14*, 1 (2007), 3–31.

[4] ALUMUR, S., AND KARA, B. Network hub location problems: The state of the art. *European Journal of Operational Research 190*, 1 (2008), 1–21.

[5] ANANDALINGAM, G., AND FRIESZ, T. Hierarchical optimization: an introduction. *Annals of Operations Research 34*, 1 (1992), 1–11.

[6] ANGELO, J., AND BARBOSA, H. A study on the use of heuristics to solve a bilevel programming problem. *International Transactions in Operational Research* (2015), 1–22.

[7] ARROYO, J., AND FERNÁNDEZ, F. A genetic algorithm for power system vulnerability analysis under multiple contingencies. *in Talbi (Ed),Metaheuristics for Bi-level Optimization* (2013), 41–68.

[8] AUSIELLO, G., PROTASI, M., MARCHETTI-SPACCAMELA, A., GAMBOSI, G., CRESCENZI, P., AND KANN, V. Complexity and approximation: combinatorial optimization problems and their approximability properties. *Springer, Berlin* (1999).

[9] BARD, J. Naval research logistics quartely. *Wiley Online Library* (1984).

[10] BARD, J. Some properties of the bilevel programming problem. *Journal of Optimization Theory and Applications 68*, 2 (1991).

[11] BARD, J. F. Practical bilevel optimization: Algorithms and applications. *Dordrecht, Kluwer Academic Publishers* (1998).

[12] BAZARAA, M. S., JARVIS, J. J., AND SHERALI, H. D. Linear programming and network flows. *Jhon Wiley & Sons Inc.* (1990).

[13] BEN-AYED, O., AND BLAIR, C. Computational difficulties of bilevel linear programming. *Operations Research 38*, 3 (1990), 556–560.

[14] BERESNEV, V. Upper bounds for objective functions of discrete competitive facility location problems. *Journal of Applied and Industrial Mathematics 3*, 4 (2009), 419432.

[15] BERESNEV, V. Local search algorithms for the problem of competitive location of enterprises. *Automation and Remote Control 73*, 3 (2012), 425439.

[16] BHADURY, J., JARAMILLO, J., AND BATTA, R. On the use of genetic algorithms for location problems. *Computers and Operations Research 29*, 1 (2002), 761–779.

[17] BLUM, C. Hybrid metaheuristics. *Computers & Operations Research 37*, 3 (2010), 430–431.

[18] BOLAND, N., DOMÍNGUEZ-MARÍN, P., NICKEL, S., AND PUERTO, J. Exact procedures for solving the discrete ordered median problem. *Computers & Operations Research 33*, 1 (2006), 32703300.

[19] CALVETE, H., GALÉ, C., AND IRANZO, J. An efficient evolutionary algorithm for the ring star problem. *European Journal of Operational Research 231*, 1 (2013), 2233.

[20] CALVETE, H., GALÉ, C., AND MATEO, P. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research 188*, 1 (2008), 14–28.

[21] CAMACHO-VALLEJO, J., CASAS-RAMÍREZ, M., AND MIRANDA, P. The p-median bilevel problem under preferences of the customers. *In R.Z. Ríos-Mercado et al. (Eds.), Recent Advances in Theory, Methods and Practice of Operations Research (pp. 121-127). Monterrey : UANL-Casa Universitaria del Libro* (2014).

[22] CAMACHO-VALLEJO, J., CORDERO-FRANCO, A., AND GONZÁLEZ-RAMÍREZ, R. Solving the bilevel facility location problem under preferences by a stackelberg-evolutionary algorithm. *Mathematical Problems in Engineering, Article ID 430243, 14 pages* (2014).

[23] CAMACHO-VALLEJO, J., MAR-ORTIZ, J., LÓPEZ-RAMOS, F., AND RODRÍGUEZ, R. A genetic algorithm for the bi-level topological design of local area networks. *PLoS ONE 10*, 6 (2015).

[24] CAMPBELL, J. Hub location and the p-hub median problem. *Operations Research 44*, 6 (1996), 923–935.

[25] CANDLER, W., AND NORTON, R. Multi-level programming and development policy. *Working paper No. 258, World Bank, Washington DC, United States* (1977).

[26] CÁNOVAS, L., GARCÍA, S., LABBÉ, M., AND MARÍN, A. A strengthened formulation for the simple plant location problem with order. *Operations Research Letters 35*, 2 (2007), 141–150.

[27] CARAMIA, M., AND MARI, R. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters, DOI: 10.1007/s11590-015-0918-z* (2015).

[28] CATTRYSSE, D., AND VAN WASSENHOVE, L. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research 60*, 1 (1992), 260–272.

[29] CHURCH, R., AND REVELLE, C. The maximal covering location problem. *Papers of the Regional Science Association 32*, 1 (1974), 101–118.

[30] COLOMÉ, R., AND SERRA, D. Consumer choice in competitive location models: formulations and heuristics. *Papers in Regional Science 80*, 4 (2001), 439–464.

[31] COLSON, B., MARCOTTE, P., AND SAVARD, G. An overview of bilevel optimization. *Annals of Operations Research 153*, 1 (2007), 235–256.

[32] CONTRERAS, I. Hub location problems. *Location science* (2015), 311–344.

[33] CONTRERAS, I., AND DÍAZ, J. Scatter search for the single source capacitated facility location problem. *Annals of Operations Research 157*, 1 (2008), 73–89.

[34] DASKIN, M. Network and discrete location: models, algorithms and applications. *Wiley-Intersciene Publication John Wiley & Sons Inc.* (1995).

[35] DE BOER, P., KROESE, D., MANNOR, S., AND RUBINSTEIN, R. A tutorial on the cross-entropy method. *Annals of Operations Research 134*, 1 (2005), 19–67.

[36] DELMAIRE, H., DÍAZ, J., FERNÁNDEZ, E., AND ORTEGA, M. Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *INFOR:Information Systems and Operational Research 37*, 3 (1999), 194–225.

[37] DEMPE, S. A necessary and a sufficient optimality condition for bilevel programming problems. *Optimization 25*, 4 (1992), 341–354.

[38] DEMPE, S. Foundations of bileven programming. *Kluwer Academic Publishers New York, United States* (2002).

[39] DEMPE, S. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization 52*, 3 (2003), 333–359.

[40] DÍAZ, J., AND FERNÁNDEZ, E. Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research 169*, 2 (2006), 570–585.

[41] DÍAZ, J. A., LUNA, D. E., AND ZETINA, C. A. A hybrid algorithm for the manufacturing cell formation problem. *Journal of Heuristics 19*, 1 (2013), 77–96.

[42] DIBBIE, C., AND DENSHAM, P. Generating intersecting alternatives in gis and sdss using genetic algorithms. *In: GIS/LIS Symposium, Lincoln.* (1993).

[43] DREZNER, Z., AND HAMACHER, H. Facility location: Applications and theory. *Berlin, Springer-Verlag* (2002).

[44] DUARTE, A., AND MARTÍ, R. Tabu search and grasp for the maximum diversity problem. *European Journal of Operational Research 178*, 1 (2007), 71–84.

[45] EISELT, H., AND LAPORTE, G. Sequential location problems. *European Journal of Operational Research 96*, 2 (1996), 217–231.

[46] EISELT, H., LAPORTE, G., AND THISSE, J. Competitive location models: a framework and bibliography. *Transportation Science 27*, 1 (1993), 4454.

[47] EISELT, H., AND MARIANOV, V. Foundations of location analysis. *New York, Springer* (2011).

[48] ESPEJO, I., MARÍN, A., AND RODRÍGUEZ-CHÍA, A. Closest assignment constraints in discrete location problems. *European Journal of Operational Research 219*, 1 (2012), 49–58.

[49] FARAHANI, R., ASGARI, N., HEIDARI, N., HOSSEININIA, M., AND GOH, M. Covering problems in facility location: A review. *Computers and Industrial Engineering 62*, 1 (2012), 368–407.

[50] FEO, T., AND RESENDE, M. Greedy randomized adaptative search procedures. *Journal of Global Optimization 6*, 1 (1995), 109–134.

[51] FERONE, D., FESTA, P., AND RESENDE, M. Hybridizations of grasp with path relinking for the far from most string problem. *International Transactions in Operational Research 23*, 3 (2016), 481506.

[52] FISHER, K. Sequential discrete p-facility models for competitive location planning. *Annals of Operations Research 111*, 1 (2002), 253–270.

[53] FONTAINE, P., AND MINNER, S. Benders decomposition for discrete continuous linear bilevel problems with application to traffic network design. *Transportation Research Part B 70*, 1 (2014), 163–172.

[54] FRANCIS, R., LOWE, T., RAYCO, M., AND TAMIR, A. Aggregation error for location models: survey and analysis. *Annals of Operations Research 167*, 1 (2009), 171–208.

[55] GALLO, M., D'ACIERNO, L., AND MONTELLA, B. A meta-heuristic approach for solving the urban network design problem. *European Journal of Operational Research 201*, 1 (2010), 144–157.

[56] GALVÃO, R. Uncapacitated facility location problems: contributions. *Pesquisa Operacional 24*, 1 (2004), 7–38.

[57] GARCÍA, S., AND MARÍN, A. Covering location problems. *In G. Laporte, S. Nickel, F. Saldanha da Gama (Eds.), Location Science, Springer International Publishing, New York, United States* (2015), 93–114.

[58] GARCÍA-LÓPEZ, F., MELIÁN-BATISTA, B., MORENO-PÉREZ, J., AND MORENO-VEGA, J. Parallelization of the scatter search for the p-median problem. *Parallel Computing 29*, 1 (2003), 575–589.

[59] GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research 13*, 5 (1986), 533–549.

[60] GLOVER, F. Tabu search - part i. *ORSA Journal of ComputinG 1*, 3 (1989), 190–206.

[61] GLOVER, F. Tabu search - part ii. *ORSA Journal of Computing 2*, 1 (1990), 4–32.

[62] GLOVER, F., KELLY, J. P., AND LAGUNA, M. Genetic algorithms and tabu search: hybrids for optimization. *Computers & Operations Research 22*, 1 (1995), 111–134.

[63] GONZÁLEZ, X., RAMÍREZ, J., MARMOLEJO, J., AND CAICEDO, G. Methodology for multi-area state estimation solved by a decomposition method. *Electric Power Systems Research 123*, 1 (2015), 92–99.

[64] HAKIMI, S. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research 12*, 3 (1964), 450–459.

[65] HAKIMI, S. On locating new facilities in a competitive environment. *European Journal of Operational Research 12*, 1 (1983), 29–35.

[66] HANJOUL, P., AND PEETERS, D. Facility location problem with clients' preference orderings. *Regional Science and Urban Economics 17*, 3 (1987), 451–473.

[67] HANSEN, P., KOCHETOV, Y., AND MLADENOVIC, N. Lower bounds for the uncapacitated facility location problem with user preferences. *Preprint G-2004-24, Mart 2004, GERAD-HEC. Montreal, Canada* (2004).

[68] HEJAZI, S., MEMARIANI, A., JAHANSHAHLOO, G., AND SEPEHRI, M. Linear bilevel programming solution by genetic algorithm. *Computers & Operations Research 29*, 1 (2002), 1913–1925.

[69] HOLMBERG, K., RONNQVIST, M., AND YUAN, D. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research 133*, 1 (1999), 544–559.

[70] HOTELLING, H. Stability in competition. *The Economic Journal 39*, 153 (1929), 4157.

[71] KALASHNIKOV, V., DEMPE, S., PÉREZ-VALDÉZ, G., KALASHNYKOVA, N., AND CAMACHO-VALLEJO, J. Bilevel programming and applications. *Mathematical Problems in Engineering, In Press* (2015).

[72] KALCSICS, J., NICKEL, S., PUERTO, J., AND RODRÍGUEZ-CHÍA, A. The ordered capacitated facility location problem. *TOP 18*, 1 (2009), 203222.

[73] KARIV, O., AND HAKIMI, S. An algorithmic approach to network location problem. part ii: The p-medians. *SIAM Journal on Applied Mathematics 37*, 1 (1979), 539–560.

[74] KESKIN, B., AND ÜSTER, H. A scatter search-based heuristic to located capacitated transshipment point. *Computers & Operations Research 34*, 1 (2007), 312–3125.

[75] KRARUP, J., AND PRUZAN, P. The simple plant location problem: survey and synthesis. *European Journal of Operational Research 12*, 1 (1983), 36–81.

[76] KRESS, D., AND PESCH, E. Sequential competitive location on networks. *European Journal of Operational Research 217*, 1 (2012), 483–499.

[77] KÜCÜKAYDIN, H., ARAS, N., AND ALTINEL, K. A leader-follower game in competitive facility location. *Computers & Operations Research 39*, 1 (2012), 437–448.

[78] KUEHN, A., AND HAMBURGER, M. A heuristic program for locating warehouses. *Management Science 9*, 1 (1963), 643–666.

[79] LAGUNA, M., DUARTE, A., AND MARTÍ, R. Hybridizing the cross-entropy method: An application to the max-cut problem. *Computers & Operations Research 36*, 1 (2009), 487–498.

[80] LAGUNA, M., AND GONZÁLEZ-VELARDE, J. L. A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing 2*, 4 (1991), 253–260.

[81] LAGUNA, M., AND MARTÍ, R. Scatter search. *Kluwer Academic Publishers, Boston* (2003).

[82] LEE, J., AND LEE, Y. Facility location and scale decision problem with customer preference. *Computers & Industrial Engineering 63*, 1 (2012), 184–191.

[83] LI, H., AND WANG, Y. A genetic algorithm for solving a special class of nonlinear bilevel programming problems. *Lecture Notes in Computer Science 4490*, 1 (2007), 1159–1162.

[84] LI, H., AND WANG, Y. An evolutionary algorithm with local search for convex quadratic bilevel programming problems. *Applied Mathematics and Information Sciences 5*, 1 (2011), 139–146.

[85] LIM, A., AND WANG, F. A smoothed dynamic tabu search embedded grasp for m-vrptw. *In 16th IEEE International Conference on Tools with Artificial Intelligence* (2004), 704–708.

[86] LIU, B. Stackelberg-nash equilibrium for multilevel programming with multiple followers using genetic algorithms. *Computer Mathematics Application 36*, 1 (1998), 79–89.

[87] LIU, Q., CAI, W., SHEN, J., FU, Z., LIU, X., AND LINGE, N. A speculative approach to spatial temporal efficiency with multiobjective optimization in a heterogeneous cloud environment. *Security and Communication Networks 9*, 17 (2016), 4002–4012.

[88] LOZANO, M., AND GARCÍA-MARTÍNEZ, C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. *Computers& Operations Research 37*, 3 (2010), 481–497.

[89] MARCOTTE, P. Network design problem with congestion effects: a case of bilevel programming. *Mathematical Programming 43*, 1 (1986), 142–162.

[90] MARCOTTE, P., MERCIER, A., SAVARD, G., AND VERTER, V. Toll policies for mitigating hazardous materials transport risk. *Transportation Science 43*, 1 (2009), 228–243.

[91] MARIANOV, V., AND SERRA, D. Location problem in the public sector. *In Drezner Z., Hamacher H., (eds) Facility location: applications and theory, Berlin, Springer* (2002).

[92] MARIĆ, M., STANIMIROVIĆ, Z., AND MILENKOVIĆ, N. Metaheuristic methods for solving the bilevel uncapacitated facility location problem with clients preferences. *Electronic Notes in Discrete Mathematics 39*, 1 (2012), 43–50.

[93] MARIĆ, M., STANIMIROVIĆ, Z., MILENKOVIĆ, N., AND DJENIĆ, A. Metaheuristic approaches to solving large-scale bilevel uncapacitated facility location problem with clients' preferences. *Yugoslav Journal of Operations Research 25*, 2 (2015), 361–378.

[94] MARMOLEJO-SAUCEDO, J., AND RODRÍGUEZ-AGUILAR, R. Short-term generation planning by primal and dual decomposition techniques. *Dyna 82*, 191 (2015), 58–62.

[95] MARTÍ, R., CORBERÁN, A., AND PEIRÓ, J. Scatter search for an uncapacitated p-hub median problem. *Computers & Operations Research 58*, 1 (2015), 53–66.

[96] MATHIEU, R., PITTARD, L., AND ANANDALINGAM, G. Genetic algorithm based approach to bilevel lineal programming. *Renue francaise d'automatique, dnformatique et de recherche opérationnelle. Recherche Opérationnelle 28*, 1 (1994), 1–21.

[97] MIRCHANDANI, P. The p-median problem and generalizations. *In Mirchandani P.B., Francis R.L. (eds) Discrete location theory, New York, Wiley* (1990).

[98] MLADENOVIC, N., BRIMBERG, J., HANSEN, P., AND MORENO-PÉREZ, J. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research 179*, 3 (2007), 927–939.

[99] MOORE, J., AND BARD, J. The mixed integer linear bilevel programming problem. *Operations Research 38*, 5 (1990), 911–921.

[100] NICKEL, S. Discrete ordered weber problems. *Operations Research Proceedings, Springer, Berlin* (2001).

[101] NICKEL, S., AND PUERTO, J. Location theory: a unified approach. *Springer, Berlin* (2005).

[102] ODUGUWA, V., AND ROY, R. Bilevel optimisation using genetic algorithm. *Proceedings IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002) 32*, 1 (2002).

[103] PESSOA, L., RESENDE, M., AND RIBEIRO, C. A hybrid lagrangean heuristic with grasp and path-relinking for set k-covering. *Computers & Operations Research 40*, 12 (2013), 31323146.

[104] PLASTRIA, F. Static competitive facility location: An overview of optimisation approaches. *European Journal of Operational Research 129*, 3 (2001), 461–470.

[105] POTVIN, J., AND ROUSSEAU, J. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research 66*, 3 (1993), 331–340.

[106] PRAIS, M., AND RIBEIRO, C. Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing 12*, 3 (2000), 164–176.

[107] PUERTO, J., AND FERNÁNDEZ, F. The symmetrical single facility location problem. *Technical report, Faculty of Mathematics, University of Sevilla* (1995).

[108] PUERTO, J., RAMOS, A., AND RODRÍGUEZ-CHÍA, A. Single-allocation ordered median hub location problems. *Computers & Operations Research 38*, 2 (2011), 559–570.

[109] PUERTO, J., RAMOS, A., RODRÍGUEZ-CHÍA, A., AND SÁNCHEZ-GIL, M. Ordered median hub location problems with capacity constraints. *Transportation Research Part C: Emerging Technologies* (2015).

[110] RAHMANI, A., AND MIRHASSANI, S. Lagrangean relaxation-based algorithm for bi-level problems. *Optimization Methods and Software 30*, 1 (2015), 1–14.

[111] RANJBAR, M., KIANFAR, F., AND SHADROKH, S. Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation 196*, 2 (2008), 879–888.

[112] REESE, J. Solution methods for the p-median problem: an annotated bibliography. *Networks 48*, 1 (2006), 125–142.

[113] RESENDE, M. Computing approximate solutions of the maximum covering problem with grasp. *Journal of heuristics 4*, 1 (1998), 161–177.

[114] RESENDE, M., AND GONZÁLEZ-VELARDE, J. Grasp: Procedimientos de búsqueda miopes aleatorios y adaptativos. *Revista Iberoamericana de Inteligencia Artificial 19*, 1 (2003), 61–76.

[115] RESENDE, M., RIBEIRO, C., GLOVER, F., AND MARTÍ, R. Scatter search and path-relinking: Fundamentals, advances, and applications. *In Handbook of metaheuristics,Springer, USA* (2010), 87–107.

[116] REVELLE, C., AND SWAIN, R. Central facilities location. *Geographical analysis, Wiley 2*, 1 (1970), 30–42.

[117] RÍOS-MERCADO, R., AND FERNÁNDEZ., E. A reactive grasp for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research 36*, 3 (2009), 755–776.

[118] ROBBINS, M., AND LUNDAY, B. A bilevel formulation of the pediatric vaccine pricing problem. *European Journal of Operational Research 264*, 1 (2016), 634–645.

[119] RUBINSTEIN, R. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability* (1999), 127–190.

[120] RUSSELL, S., AND NORVIG, P. Artifitial intelligence: A modern approach. *New Jersey, United States: Prentice Hall* (2010).

[121] SAHARIDIS, G., AND LERAPETRITOU, M. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization 44*, 1 (2009), 29–51.

[122] SAHIN, G., AND SÜRAL, H. A review of hierarchical facility location models. *Computers & Operations Research 34*, 8 (2007), 2310–2331.

[123] SCHEUERER, S., AND WENDOLSKY, R. A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research 169*, 1 (2006), 533–547.

[124] SCHILLING, D., JAYARAMAN, V., AND BARKHI, R. A review of covering problem in facility location. *Computers & Operations Research 1*, 1 (1993), 25–55.

[125] SERRA, D., AND REVELLE, C. Market capture by two competitors: the preemptive location problem. *Journal of Regional Science 34*, 4 (1994), 549561.

[126] SHI, X., LIANG, Y., LEE, H., LU, C., AND WANG, L. An improved ga and a novel pso-ga-based hybrid algorithm. *Information Processing Letters 93*, 5 (2005), 255–261.

[127] SINHA, A., MALO, P., AND DEB, K. An improved bilevel evolutionary algorithm based on quadratic approximations. *Proceeding of the 2014 IEEE Congress on Evolutionary Computation* (2014), 18701877.

[128] SINHA, A., MALO, P., AND DEB, K. Solving optimistic bilevel programs by iteratively approximating lower level optimal value function. *Proceeding of the 2016 IEEE Congress on Evolutionary Computation* (2016), 1877–1884.

[129] SINHA, A., MALO, P., AND DEB, K. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research 257*, 2 (2017), 395–411.

[130] TALBI, E. G. A taxonomy of hybrid metaheuristics. *Journal of Heuristics 8*, 5 (2002), 541–564.

[131] TANSEL, B., FRANCIS, R., AND LOWE, T. Location on networks: a survey. part i: the p-center and p-median problems. *Management Science 29*, 1 (1983), 482–497.

[132] TEITZ, M., AND BART, P. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research 16*, 1 (1968), 955961.

[133] TEXEIRA, J., AND ANTUNES, A. A hierarchical location model for public facility planning. *European Journal of Operational Research 185*, 1 (2008), 92–104.

[134] VALLADA, E., AND RUIZ, R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega 38*, 1 (2010), 57–67.

[135] VASIL'EV, I., AND KLIMENTOVA, K. The branch and cut method for the facility location problem with clients preferences. *Journal of Applied and Industrial Mathematics 4*, 3 (2010), 441–454.

[136] VASIL'EV, I., KLIMENTOVA, K., AND KOCHETOV, Y. New lower bounds for the facility location problem with clients preferences. *Computational Mathematics and Mathematical Physics 49*, 6 (2009), 1010–1020.

[137] VICENTE, L., AND CALAMAI, P. Bilevel and multilevel programming: a bibliography review. *Journal of Global Optimization 5*, 1 (1994), 291306.

[138] WANG, G., WAN, Z., WANG, X., AND LV, Y. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications 56*, 1 (2008), 2550–2555.

[139] WEBER, A. Theory of the location of industries. *University of Chicago Press, Chicago, USA* (1929).

[140] YE, J., AND ZHU, D. L. Optimality conditions for bilevel programming problems. *Optimization 33*, 1 (1995), 9–27.

[141] ZENG, B., AND AN, Y. Solving bilevel mixed integer program by reformulations and decomposition. *Technological report, Dept. of Industrial and Management Systems Engineering, University of South Florida* (2014).