INTERNATIONAL JOURNAL OF
INTELLIGENT OPTIMIZATION
AND LOGISTICS

Available online at www.ijiol.com

RESEARCH ARTICLE

A Golden Ball Algorithm for the Multi-Level Multi-Capacitated Facility Location Problem: A Comparative Study with Genetic Algorithm

Khalil Abbal [a*]

[a] Smart Systems Laboratory, ENSIAS, Mohammed V University in Rabat, Morocco

ABSTRACT

This study addresses the Multi-Level Multi-Capacitated Facility Location Problem (ML-MCLP), which involves determining facility locations, capacity levels, and customer demands across a multi-level network while minimizing total assignment system costs. Given the NP-hard nature of the problem, we proposed a problem-specific "Golden Ball" (GBA) algorithm that incorporates a custom solution representation, feasibility checking mechanisms, and local improvement procedures. A comprehensive computational study is conducted on a diverse set of instances covering different problem sizes and capacity levels. Small and medium-sized instances are also solved using an exact mixed-integer programming solver to provide optimal or reference solutions. The proposed approach is compared to the genetic algorithm (GA) in terms of solution quality, computation time, robustness, and scalability. Several independent runs and statistical tests are used to evaluate the differences in performance. The results show that the proposed approach, based on the GBA algorithm, provides high-quality solutions and offers an optimal balance between solution quality and computational effort, particularly for large instances. These results demonstrate the potential of problem-specific GBA mechanisms to efficiently solve large-scale ML-MCLP instances.

KEYWORDS Facility location · multi-capacity · multi-level · genetic algorithm · golden ball algorithm

1. Introduction

Ensuring the effective functioning of any business or public organization, it is critical to provide consumers with uninterrupted access to the services provided. Companies are seeking to enhance customer demand for their services by effectively embedding their services in supply chains, improving customer satisfaction, and lowering operating costs. In the same way, facility location problems (FLP) play a key role in optimizing critical government services such as education, health care, or emergency relief that are oriented in space. In designing these systems, there are two vital elements that play a prominent role namely, the selection of suitable facilities within a supply chain and the product flow management across all system levels. Nevertheless, the decision process usually fails to consider factors pertaining to capacity selection. The strategic or operational level decisions about capacity can be very important due to significant economic

* Corresponding author: khalil.abbal@gmail.com.
Article history: Received 15 September 2026; Revised 18 October 2026; Accepted 6 November 2026; Published 11 November 2026.
Publication details: Volume 1, Issue 1, Article 0001.

© 2026 The Authors. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0).

impacts on companies. We find the studies that include the capacity decision in the work of Abbal et al [1] [2] and El-Amrani et al [3].

Facility location problem takes a wide place in operation research. It consists of finding the best location from a set of potential facilities so as to satisfy the customer's demands. It has many important applications in real world including distribution system, supply chain and telecommunication. Many variants of the facility location have been introduced and there is an abundance of studies that deal with different variants of location problems and several approaches have been proposed to solve these problems. Especially for multi-level location problems, researchers have increasingly begun to take an interest in solving these problems and several approaches have therefore been proposed in this direction. The exact approach found in the works of Tcha et Lee. [4] that based on Branch and Bound algorithm. Ortiz et al. [5] proposed an algorithm based on Bender formulation to deal with the multi-level uncapacitated facility location problem, the approximate methods found in the works of Chardaier [6] which it developed a simulated annealing algorithm for the two-level simple plant location problem. Tra et al. [7] and Hinojoa et al. [8] developed heuristics based on lagrangian relaxation.

The GBA, amongst the metaheuristics, is a multiple population-based metaheuristic. This method has only recently been introduced by Osaba et al. [9] and has worked well in addressing numerous optimization problems. Osaba et al. [9] were the first to provide a reproducible version of GBA focusing on the combined solution of the Traveling Salesman Problem (TSP) and the Capacitated Vehicle Routing Problem (CVRP) problems. Osaba et al. [10] applied the GBA to a number of optimization problem such as the asymmetric TSP, vehicle routing problem with backhaul, n-queen problem and 1-d bin packing problem. Ruttanateerawichien et al. [11] introduced an improved version of GBA adapted specifically for solving the CVRP. Finally, Berahhou et al. [12] solved the Dynamic Vehicle Routing Problem with Simultaneous Delivery and Pickup, Overtime and Inventory by applying the GBA algorithm. To our knowledge, no previously published paper has utilized the GBA to address the Facility location problems.

The GA, introduced by John Halland in 1975 [13], has also been used successfully to solve a number of combinatorial optimization problems. For more detail about the GA algorithm a survey was presented by Katoch et al. [14]. Leu et al. [15] deals with the GA for a mixed model assembly line, Watanabe et al. [16] developing GA for the job-scheduling problem. Finally, A. Berrahou and Y. Benadada [17] uses a memetic algorithm to deal with the vehicle routing problem with simultaneous delivery and pick up. For the facility location problem, we find the application of the Genetic Algorithm in the papers of Fernandes et al. [18] and Alcaraz et al. [19] and many papers such as [20] [21] [22]. Since the GA demonstrates his effectiveness in tackling optimization problem like the FLP, we decide to apply to the ML-MCLP problem as comparative method to the GBA algorithm.

The remainder of this article is organized as follows. First, we define the ML-MCLP model and present its main characteristics as well as its mathematical formulation. Next, we present the GBA and GA algorithms, whose parameters are calibrated for our problem. A numerical study is then conducted to evaluate the performance of these two algorithms. Finally, the article concludes with the main results, recommendations, and suggestions for future research.

2. Formulation

To describe the ML-MCLP model, whose mathematical formulation was first presented by Abbal et al. [2], we will begin by outlining the system's assignment levels. Each level establishes a relationship between two entities, one acting as the customer and the other as the supplier responsible for fulfilling the customer's demand. With the exception of end customers, all other entities are referred to as "facility". This facility is characterized by set of capacity level to choose one of them. According to the end customer requirements, the system imposes the facilities to open over all the assignments level of the system and to choose the appropriate capacity for each opened facility from a predefined list. For each defined assignment level, we have a limited budget for opening the facilities.

The main decisions provided by this model are which facilities to open at each allocation level, while respecting the budget allocated to that level. The appropriate capacity to use for each facility and, finally, the flow of products through the system to meet the end-customer demands. The aim of this problem is to minimize the total assignment cost.

Figure 1 illustrates the ML-MCLP problem by a particular case of the bi-level. It is a two-level assignment system, with four customers, three facilities for the first level and two facilities for the second one. We also offer two levels of capacity for each facility.

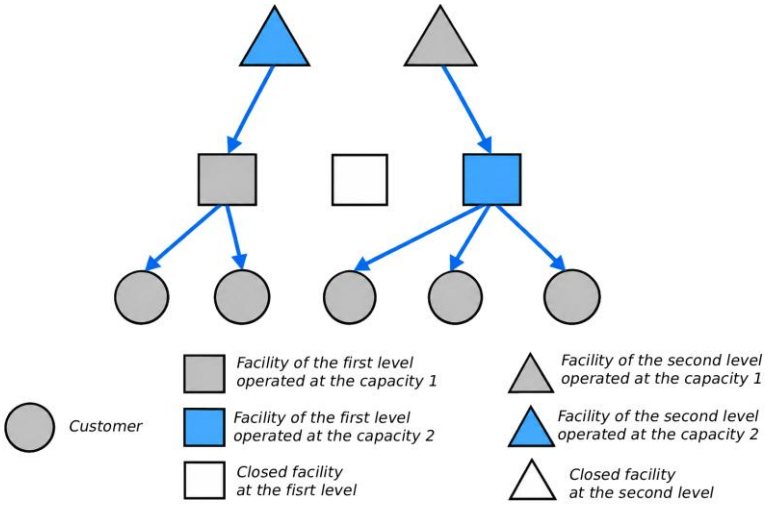


Figure 1. Graphical representation of the ML-MCLP Problem.

The ML-MCLP can be formulated as a mixed integer linear program. Let:

- L: Set of assignment levels.
- A^l: Set of edge (i, j) at the l^{th} assignment level.
- A_+^l: Set of facilities j such that (i, j) $\in A^l$.
- A_-^l: Set of facilities i such that (i, j) $\in A^l$.
- K^l: Set of capacity levels for facilities at the l^{th} assignment level.

Consider the following parameters:

- c_{ij}^l: Assignment cost of facility i to facility j at assignment level l.
- d_j: Demand of customer j.
- u_i^{lk}: Capacity of the facility i at the assignment level l, using the capacity level k.
- f_i^{lk}: Opening cost of the facility i at the assignment level l, using the capacity level k.
- B^l: Limit budget on the sum of the opening cost of facilities at the l^{th} assignment level.

Decision variables:

- $x_{ij}^l \in \mathbb{R}^+$: Quantity of product supplied by the facility i to the facility j at the assignment level l.
- $y_i^{lk} = \begin{cases} 1, & \text{if the facility i at the level l is opened and used at the capacity level k} \\ 0, & \text{otherwise} \end{cases}$

The ML-MCLP mathematical formulation is as follows:

$$\boldsymbol{Min} \sum_{l=1}^{L} \left(\sum_{(i,j)\in A^l} c_{ij}^l \, x_{ij}^l \right) \tag{1}$$

$$\sum_{i \in A_-^1} x_{ij}^1 = d_j \,, \qquad \forall \, j \in A_+^1 \tag{2}$$

$$\sum_{j \in A_+^l} x_{ij}^l \leq \sum_{k \in K^l} u_i^{lk} y_i^{lk} \,, \forall \, l \in L \,, \forall i \in A_-^l \tag{3}$$

$$\sum_{k \in \mathrm{K}^l} y_i^{lk} \leq 1 \quad , \quad \forall\, l \in \mathrm{L}, \forall\, \mathrm{i} \in A_-^l \tag{4}$$

$$\sum_{\mathrm{i} \in A_-^l} x_{ij}^l \geq \sum_{\mathrm{p} \in A_+^{l-1}} x_{jp}^{l-1} \quad , \quad \forall \mathrm{j} \in A_+^l, \forall\, l \in L \backslash \{1\} \tag{5}$$

$$\sum_{i \in A_-^l} \sum_{k \in \mathrm{K}^l} f_i^{lk} y_i^{lk} \leq \mathrm{B}^l \quad , \quad \forall\, l \in L \tag{6}$$

$$x_{ij}^l \in \mathbb{R}^+ \quad , \qquad \forall\, l, \forall\, (i,j) \in \mathrm{A}^l \tag{7}$$

$$y_i^{lk} \in \{0,1\} \quad , \qquad \forall\, l \in L \,, \forall\, k \in \mathrm{K}^l \tag{8}$$

The objective function (1) aims to minimize the total assignment cost. The latter does not only depend on the distances between the facilities but also on the quantity of the transported flow. Constraints (2) guarantee that the demand of each customer is entirely satisfied. Constraints (3) define the capacities to be respected in each assignment level of the system. Indeed, at the first level, facilities provide enough resources for the demands actually expressed in the data instance. However, all other capacity constraints are primarily technical constraints, as their demands are calculated by the system. Constraints (4) prohibit the opening of facilities over all assignment level to at less one level of capacity. Constraints (5) guarantee the flow balance of the product over the facilities. (6) are the budget constraints. Finally, constraints (7) and (8) define the bounds and types of decision variables.

The ML-MCLP formulation, presented in this section, is of primary importance for the modeling of real situations. It generalizes the location problems to the case of a multi-level system with several capacity levels. It is also presented in a compact and elegant format, based on the customer-facility principle in each level. Given the NP-hard nature of the ML-MCLP, as it is an extension of the NP-hard problem (BMCLP), solving its formulation by exact methods is time-consuming for large problems. The metaheuristics are an alternative to provide very good solutions in a reasonable time. In the following section, we will apply The GBA and the GA to the ML-MCLP problem.

3. Genetic algorithm

The GA is a metaheuristic introduced by John Halland; it is used to solve hard optimization problems. It represents a part of the field called evolutionary computation, which imitates the natural selection proposed by Darwin. The GA works with a set of solutions called population and evolves them by applying, at each iteration, the GA operators, such as crossover and mutation. In the following, we introduce the main components of the GA to solve the ML-MCLP.

3.1. Solution representation and Initial population

An important point in designing a GA algorithm is a representation of a solution. Several coding variants have been used to solve combinatorial problems as in papers [23] and [24]. In our approach, if we fix the variable y_i^{lk} corresponding to the opening of the facility i at the assignment level l used at the capacity level k, it remains easy to find the flow of the product x_{ij}^l over all levels of the system. Thus, we will adopt a representation similar to the binary representation for our problem, each one contents $\sum_1^L |A_1^-|$ bits, each bit can contain a random capacity value from the predefined capacity list if the corresponding facility is open, otherwise it equals zero. For the initial population, we initialize a set of feasible solutions for the ML-MCLP problem, which are randomly generated.

As example, consider a system with two assignment level and two capacity level for each facility. The first level contains four facilities and the second three. Specifically, all facilities with higher levels are represented before facilities of the lower level. The following table lists the capacities of each facility that will be adopted in the following paragraph. The Figure 2 and Table 1 represents a chromosome with two levels.

Table 1: Example of the capacity level for the facilities.

	Facilities of the second level			**Facilities of the first level**			
Second capacity level	22	23	27	38	47	32	55
First capacity level	29	25	43	43	57	57	24

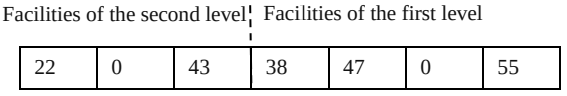


Figure 2. Representation of a chromosome

3.2. Fitness value

The computation of the fitness is the most time-consuming part of calculation in the GA, because it requires an estimation for all individuals in the population. Generally, an efficient method for calculating the fitness value of a function, can inversely influence the execution time of the algorithm. In our case, for each chromosome we obtain the following sub-problem:

$$\boldsymbol{Min} \sum_{\mathrm{l}=1}^{L} \left(\sum_{(i,j)\in \mathrm{A}^{l}} c_{ij}^{l}\, x_{ij}^{l} \right) \tag{9}$$

$$\sum_{i\in A_{-}^{1}} x_{ij}^{1} = d_j \,, \forall\, \mathrm{j} \in A_{+}^{1} \tag{10}$$

$$\sum_{\mathrm{j}\in A_{+}^{l}} x_{ij}^{l} \leq C_i^{l} \quad , \forall\, l \in \mathrm{L} \,, \forall \mathrm{i} \in A_{-}^{l} \tag{11}$$

$$\sum_{\mathrm{i}\in A_{-}^{l}} x_{ij}^{l} \geq \sum_{\mathrm{p}\in A_{+}^{l-1}} x_{jp}^{l-1} \,, \forall \mathrm{j} \in A_{+}^{l}, \forall\, l \in L\backslash\{1\} \tag{12}$$

$$x_{ij}^{l} \in \mathbb{R}^{+} \,, \forall\, l, \forall\, (i,j) \in \mathrm{A}^{l} \tag{13}$$

Where the C_i^lcorrespond to the predefined capacity in the chromosome for the i facility at the l assignement level l. Considering the above sub-problem, we propose to calculate the fitness value taking into account that each assignment level can be considered as a customer-supplier relationship, then starting from the lowest to the highest assignment level we begin by iteratively assigning open facilities, with the corresponding capacity according to the chromosome, to the nearest customers while satisfying the capacity constraint.

3.3. GA Operators

In each iteration of the GA, a new population is created from the fittest individuals of the current and the new generations. In our case, we keep the best chromosome for every generation as the best candidate for all generations, and other individuals' chromosomes were selected by roulette wheel selection, based on probabilities of fitness of parents and children. The two crossover points applied in this study, whereby one is randomly chosen in the first assignment level and another in the second assignment level. This approach creates two offsprings with different features from both parents and this increases the level of variation in the population. The mutation operator complements the directional solution-space search and also enhances the degree of convergence. This operator works by reintroducing some elements of chance into the population and thus encourages diversity which might inhibit premature convergence. For discrete problems mutation is a process of selecting one gene in one chromosome at random and substituting it with a gene of a random value. This encourages diversity in the population which also appreciates effective search without the need for total search.

3.4. Post check Operators

During the iteration of the GA, we can have children representing non feasible solutions for our problem. To remedy this constraint, we incorporated in the GA a procedure of post-check allowing to verify the feasibility of the children created while respecting the constraints of capacity and budget. The post-check adjusts the created children by adding the capacities for the opened facilities or by adding an open facility in order to create feasible children.

4. Golden ball algorithm

Based on the principles of soccer, the GBA is a metaheuristic that has a multi-population, which uses different football-related concepts for the search process. First, GBA starts by creating the initial population, then divides its solutions (called players) into several sub-populations (called teams) of the system. Once this initial phase is complete, the first season begins. A season is divided into weeks, during which teams train and compete against each other, creating a competitive league. At the end of the season, the transfer phase begins,

Algorithm 1: Pseudo-code of the GBA

Create the initial population
Divide the population into multiple teams (T)
Repeat
  Restart the competition league
    For each match **do**
      For each team **do**
        Perform conventional training
      Perform custom training
      Calculate the quality of each player
      End For
    Play the game
  End For
  Transfer players between teams
Until the stopping criterion is met
Return the best player and its corresponding quality

during which players and coaches can change teams. This process is repeated until the end criterion is met. In the following, we introduce the algorithm for the GBA and their main components applied to solve the ML-MCLP. For further information about the GBA, refer to the paper of Osaba et al.[9].

4.1. Initialization and division into many teams

In this step, we initialize a set of feasible solutions for the ML-MCLP problem that are randomly generated, denoted as P. Each solution is represented as a player p_i where P = $\{p_1,p_2,\ldots,p_n\}$. These players are then randomly assigned to teams t_s. Following the assignment, each player is designated by a variable p_{ij} indicating player p_j of teamt_i. The collection of teams is represented as T, where T= $\{t_1, t_2 ,\ldots,t_s\}$. In our methodology, when we set the variable y_i^{lk} to represent the activation status of facility i at assignment level l, it simplifies the determination of product flow x_{ij}^{l} across all system levels through efficient sub-problem solving. Consequently, our strategy employs an encoding like used for the GA algorithm.

4.2. Players Power

The calculation of fitness values q_{ij}, for each player p_{ij}, is done by applying the same method applied in the GA algorithm. This approach ensures a comprehensive evaluation of player fitness, integrating both stochastic elements and rigorous optimization techniques for robust and accurate assessment.

4.3. Training players

The purpose of this stage is to enhance the performance of each team through a series of training sessions. We categorize these sessions into two types: conventional training and custom training.

4.3.1. Conventional trainings

This type of training is carried out regularly during the season, with the aim of improving the quality of the players. Each team has its own conventional training; therefore, each player p_{ij}, on the team is trained according their assigned method, and a new player p'_{ij} is introduced and replaced him if $q'_{ij} < q_{ij}$.

It's important to note that each training session concludes when its termination criterion is met, signaling that the quality of the coached player is no longer improving after several successor functions have been applied.

According to the nature of our problem, we can distingue different types of conventional training that represents different strategy or movements that can be employed to explore the neighborhood structure in search of better solutions. These movements are typical used in local search algorithms such as the swap

movements, which mean to swap the value of two facilities or the mutation operators applied in the GA algorithm.

4.3.2. Custom training

This training method differs from the previous one in that it doesn't occur on a regular basis; rather, it is implemented only when a player becomes trapped in a local optimum. This situation arises when the player undergoes numerous conventional training sessions without being able to improve its quality. The custom training happens between the trapped player p_{ij} and the captain p_{icap} leading to the creation of a player p_{ij}', similar to crossover operators in Genetic Algorithms. The objective of custom training is to encourage the player to explore more of the solution space.

4.4. Match playing

In this phase, each team competes against every other team in matches. A match is played by two participating teams. All the players of both teams are sorted in terms of their quality. This is followed by the comparison of the two highest-ranked players from each of the two competing teams (p_{1j} andp_{2j}). The winning of the match earns one point for the relevant team. After all players have been compared, the winner at such game is the team that has the maximum number of points. The team which wins gets 3 additional points while the defeated one gets zero point. One point is awarded to both teams in the event of a draw. Then the order is how the teams are arranged based on their total points. Refer to Table 2 for the game procedure.

Table 2. Example of playing game.

T1	*Score*	*T2*
$\boldsymbol{p_{11}}$ *Solution=12 300*	*0-1*	p_{21} *Solution=11 700*
$\boldsymbol{p_{12}}$ *Solution=15 300*	*0-2*	p_{22} *Solution=12 300*
$\boldsymbol{p_{13}}$ *Solution=17 800*	*1-2*	p_{23} *Solution=18 600*
$\boldsymbol{p_{14}}$ *Solution=32 300*	*1-3*	p_{24} *Solution=22 300*
Lose		*Win*

4.5. Players transfer

The objective of this phase is to make sure that the teams in the upper half of the league will be more reinforced with better players, whereas the teams in the lower half will have to manage the teams with such players who are not that skilled. At first, the teams are listed from the highest to the lowest score. Afterwards, all the players of each team are also ordered in descending order, but now by their player quality. After, the worst performing player of the best ranked team is replaced by the best performing player of the worst ranked team. Following, the second worst player of the second-best team is interchanged with the second-best player of the second worst team. Finally, as illustrated by the transition from "Before" to "After" in the Table 3, the end-of-season transfer window rebalances the teams: the lowest-performing players from the top-ranked teams (such as T1, highlighted in red) are systematically swapped with the top-performing players from the struggling teams (such as T4, also highlighted in red.

Table 3. Example of transfer players.

T1	T2	T3	T4	T1	T2	T3	T4
$\boldsymbol{p_{11}}$	p_{21}	p_{31}	p_{41}	p_{11}	p_{21}	p_{31}	p_{14}
$\boldsymbol{p_{12}}$	p_{22}	p_{32}	p_{42}	p_{12}	p_{22}	p_{23}	p_{42}
$\boldsymbol{p_{13}}$	p_{23}	p_{33}	p_{43}	p_{13}	p_{32}	p_{33}	p_{43}
$\boldsymbol{p_{14}}$	p_{24}	p_{34}	p_{44}	p_{41}	p_{24}	p_{34}	p_{44}
Before				After			

5. Computational study

The experimental study was conducted on a personal computer Dell Intel(R) Core(TM) i7 computer with 1.80 GHz CPU and 4.00 GB RAM and the GA and the GBA were coded using Python 3.10.

5.1. Instances

Knowing that ML-MCLP is a new variant of facility location problems, and, to our knowledge, there is no public datasets are available for testing our proposed algorithms there are no public datasets available to test our proposed algorithms. Therefore, we generated our own instances. We approached this objectively, by using both random values and established data generation methods from the literature. The created instances are based on the Counejol et al. [25] method and Klose [26]. By randomly varying the ML-MCLP parameters detailed in Section II, this method creates a diverse set of test instances with varying difficulty levels. The complexity of the problem largely depends on the number of nodes and the resources available, such as capacity and budget. Given the large number of potential variables, we restricted our numerical tests to two assignments levels. The following Table 4 shows the range of variation of its parameters.

Table 4. Range of the ML-MCLP Parameters.

Parameters	Range
$\boldsymbol{d_j}$	[5,35]
$\boldsymbol{c_{ij}^l}$	[20,200]
$\boldsymbol{u_i^{lk}}$ for the first level	[10,160]
$\boldsymbol{u_i^{lk}}$ for the second level	[20,700]
$\boldsymbol{f_i^{lk}}$	$randomly[0,90] + \text{randomly}[100,110] * \sqrt{u_i^{lk}}$

Using the ranges specified in Table 4, we generated the ML-MCLP problem data, including customer demand, facility capacities, and assignment costs. Table 5 provides details on the instance size (problem dimensions) and the budgets allocated for facility openings.

Table 5. Instances of the problem.

Instance	$\|A_1^+\| \times \|A_1^-\| \times \|A_2^-\|$	Capacity level ($\|B^1\| \times \|B^2\|$)	B^1	B^2	Number of variables	Number of constraints
KH_1	50×33×10	2×2	7000	8000	16586	128
KH_2			11000	11000		
KH_3		5×5	7000	8000	16930	
KH_4			11000	11000		
KH_5		10×10	7000	8000	17165	
KH_6			11000	11000		
KH_7	100×50×25	2×2	12000	15000	125150	227
KH_8			16000	20000		
KH_9		5×5	12000	15000	125375	
KH_10			16000	20000		
KH_11		10×10	12000	15000	125750	
KH_12			16000	20000		
KH_13	200×100×50	2×2	40000	40000	1000300	452
KH_14			60000	60000		
KH_15		5×5	40000	40000	1000750	
KH_16			60000	60000		
KH_17		10×10	40000	40000	1001500	
KH_18			60000	60000		
KH_19	300×150×70	2×2	50000	50000	3150440	672

KH_20			80000	80000		
KH_21		5×5	50000	50000	3151100	
KH_22			80000	80000		
KH_23		10×10	50000	50000	3152200	
KH_24			80000	80000		
KH_25	400×200×90	2×2	80000	80000	7200580	892
KH_26			100000	100000		
KH_27		5×5	80000	80000	7201450	
KH_28			100000	100000		
KH_29		10×10	80000	80000	7202900	
KH_30			100000	100000		
KH_31	800×400×180	2×2	120000	120000	57601160	1782
KH_32			150000	150000		
KH_33		5×5	120000	120000	57602900	
KH_34			150000	150000		
KH_35		10×10	120000	120000	57605800	
KH_36			150000	150000		

5.2. Parameters setting

The GA algorithm requires the introduction of three parameters namely: population size, crossover rate and mutation rate. As a first step we conducted an experimental study to find the suitable parameters for our GA. The GBA algorithm requires also four key parameters: the number of players per team, the iteration count for conventional training(NICVT), the iteration count for custom training(NICUT), and the number of a season. Our initial efforts involved an experimental exploration to find the adequate value for these parameters within the GBA and GA. This exploration was conducted using the Taguchi method, a robust statistical approach renowned for its efficacy.

Taguchi method is based on two tools: Orthogonal Array (OA) and signal to noise (ratio S/N). The Taguchi method based on OA allows to reduce the number of experiments; the experimental schemes are constructed from the different levels of the parameters. In the ratio S/N, S represents the variable response signal and N is the standard deviation noise. So, the S/N ratio represents the strength of change in the response variable.

The objective of Taguchi method is to maximize the S/N ratio [27]. For the Tagushi method we adopted L16 design in Minitab software. For each level we performed 10 tests on an instance of 100 ×50 ×25, which correspond to the cardinal of the following set $|A_1^+| \times |A_1^-| \times |A_2^-|$. (see Table 6 and Table 7).

Table 6: The levels used for the Tagushi method for the GA.

GA Parameters	**Levels**			
	1	2	3	4
Population Size	20	40	60	80
Cross over Rate	0,5	0,6	0,7	0,8
Mutation Rate	0,01	0,04	0,07	0,1

Table 7: The levels used for the Tagushi method for the GBA.

GBA Parameters	Levels			
	1	2	3	4
Number of players (NP)	6	8	10	12
Number of a season (NS)	4	6	8	10
NICVT	4	8	12	16
NICUT	2	4	6	8

According to the S/N ratio diagram presented by the Tagushi method in the Figure 3 and 4, the adequate values for our GA and GBA are the maximum values in the diagrams and are presented in the Table 8 and 9.

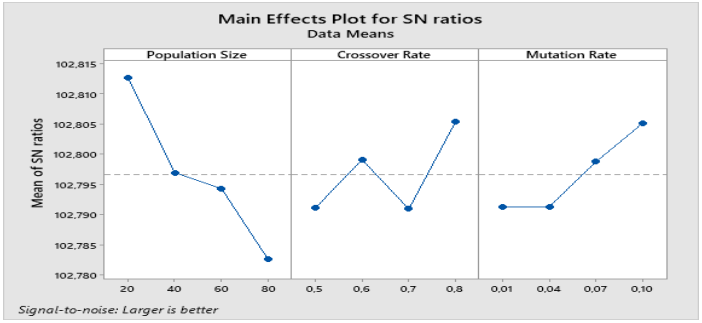


Figure 3. S/N PLOT from the Tagushi design for the GA.

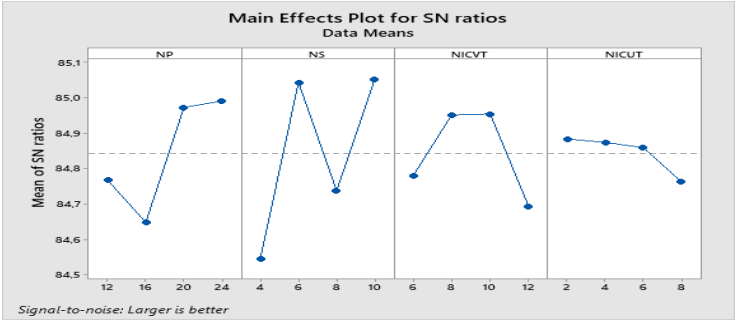


Figure 4. S/N PLOT from the Tagushi design for the GBA.

Table 8. Adequate value for the GA algorithm.

Parameters	Adequate values
Population Size	20
Crossover Rate	0,8
Mutation Rate	0,1

Table 9: Adequate value for the GBA algorithm.

Parameters	Adequate values
number of players	24
number of a season	10
NICVT	10
NICUT	2

As mentioned above, the ML-MCLP is a new variant of facility location problems, so we do not find any study concerning our problem to compare our results with it. Therefore, in order to conduct a numerical study, we decided as a first step to validate the parameter of the algorithms with the results provided by the CPLEX solver on small instances (KH_1 to KH_6). Then, we continue the study on medium and big instance (KH_7 to KH_36), to test the performances for the GBA and the GA.

5.3. Lower bounds

Given the difficulty of our problem, and in order to compare the efficiency of our algorithms on big instances, we proposed using the lower bound (LB) for large instances. The lower bound was obtained by using the Lagrangian relaxation technique that is an alternative technique for generating lower bounds, this procedure transforms a hard problem is converted into a relatively easy one by relaxing a set of complicating constraints. As a result, we proceeded to relax the budget constraint and the capacity constraint in order to find the best lower bound.

Table 10. Results of instances validation

Instance	Optimal value	LB_CAPACITY (Lower bound for relaxing the capacity constraint)	LB_BUDGET (Lower bound for relaxing the budget constraint)	GBA		GA	
				Obj	CPU (s)	Obj	CPU (s)
KH_1	30718	28242	24961	29176	7	29421	4
KH_2	29077	27825	24961	29234	7	29336	4
KH_3	30154	28322	24961	29116	7	29843	48
KH_4	29079	26799	24961	30012	7	29357	4
KH_5	28682	27782	24961	29741	15	29391	8,4
KH_6	29081	26177	24961	29120	17	29651	21

The computation study on instances (KH_1 to KH_6) is shown in the Table 10. According to the results, it is evident that the CPLEX Solver consistently delivers optimal values across all instances (KH_1 to KH_6), which designed for the validation, within a reasonable execution CPU time. The GBA and GA algorithms also show commendable performance. We conclude that both algorithms exhibit notable efficiency, producing solutions closely aligned with the optimal values within a significantly shorter CPU time. In terms of LB, relaxing the capacity constraint performs better than relaxing the budget constraint. Therefore, in our approach to test the efficiency of both algorithms for middle and big instances, we will use the lower bound obtained from relaxing the capacity constraint. We also use the gap which correspond to the deviation of the obtained solution from the GBA and GA to the LB.

Table 11. Results of the GBA and GA for Medium instances.

Instance	LB	GBA		GA		Gap GBA %	Gap GA %
		Obj	CPU (s)	Obj	CPU (s)		
KH_7	137192	137354	23	137549	10,55	0,12	0,26
KH_8	137187	137291	19,81	137689	10,03	0,08	0,37
KH_9	137560	137609	19,71	137641	10.40	0,04	0,06
KH_10	137425	137509	20,5	137528	10.70	0,06	0,07
KH_11	137360	138108	19,88	138423	10.49	0,54	0,77
KH_12	137293	137377	20	137629	10,55	0,06	0,24
KH_13	170757	181281	62	181281	51	6,16	6,16
KH_14	171128	179344	67	180588	54	4,80	5,53
KH_15	170558	180544	59	181015	54	5,85	6,13
KH_16	170666	173465	75	178325	59	1,64	4,49

KH_17	170564	174566	74	181026	49	2,35	6,13
KH_18	170650	174566	78	179722	48	2,29	5,32

Table 11 shows the results for the average instances (KH_7 to KH_18):

- Comparison of Solution Quality: GBA generally outperforms GA in terms of solution quality. In some cases, the objective values obtained by the two algorithms are very close, while in most cases, GBA provides better solutions. This advantage appears to be more pronounced for larger and more complex instances.

- Computational Efficiency: The genetic algorithm (GA) requires less CPU time than GBA in most cases, indicating that GA is generally more computationally efficient. However, this reduction in computation time is often accompanied by a lower solution quality than that of GBA.

- Comparison with the Lower Bound: Results relative to the lower bound (LB) indicate that both algorithms produce solutions close to the reference bound. They achieve an average optimality gap of only 2.3%, suggesting that both GBA and GA are capable of obtaining high-quality solutions, although GBA generally achieves better objective values.

Overall, AG tends to provide solutions more quickly, while GBA generally offers higher solution quality. The advantage of GBA becomes particularly relevant for larger and more complex instances, highlighting a trade-off between computation time and solution quality.

Table 12. Results of the GBA and GA for Big instances.

Instance	**LB**	**GBA**		**GA**		**Gap GBA %**	**Gap GA %**
		Obj	CPU (s)	Obj	CPU (s)		
KH_19	253634	261452	118	266019	88	3,08	4,88
KH_20	253577	261341	111	265908	81	3,06	4,86
KH_21	253115	261356	115	265923	95	3,26	5,06
KH_22	253280	260920	125	265487	105	3,02	4,82
KH_23	253164	261331	117	265898	87	3,23	5,03
KH_24	253204	262222	126	266789	96	3,56	5,37
KH_25	327954	331330	190	335897	160	1,03	2,42
KH_26	327994	332211	183	336778	153	1,29	2,68
KH_27	327720	332941	197	337508	167	1,59	2,99
KH_28	327785	333075	285	337642	155	1,61	3,01
KH_29	327621	331430	287	335997	157	1,16	2,56
KH_30	327665	334075	297	338642	167	1,96	3,35
KH_31	664544	666567	556	669067	450	0,3	0,68
KH_32	664475	667653	657	669853	546	0,48	0,81
KH_33	664211	667623	547	669123	488	0,51	0,74
KH_34	664150	667163	566	669663	482	0,45	0,83
KH_35	664026	667980	524	670480	423	0,6	0,97
KH_36	664017	668091	554	670591	412	0,61	0,99

For the numerical study on big instances presented in Table 12, we conclude that:

- Overall instances tested the average gap of the GBA and the GA is 1,84% and 2.93%. Considering the values of the gap and the different LB values, and take into account that the GBA give a better solution compared to the GA, we may conclude that the GBA achieve better solution quality compared to the GA.

- From the column of the CPU time of the GBA and the GA. The average CPU time for the GBA is 203 s, while the average of the GA is 172, 52 s, which indicates that the GA algorithm performs slightly better solutions than GBA algorithm in terms of execution time.

The computational study demonstrates the effectiveness of both the GBA algorithm and the GA algorithm in solving the ML-MCLP instances studied, with the GBA algorithm generally providing a higher-quality solution, while the GA algorithm requires less computation time.

6. Conclusion

In this paper, we investigated the multi-level, multi-capacity facility location problem (ML-MCLP), which aims to determine the most suitable locations and operating capacities for facilities across the system's various assignment levels, while satisfying customer demand. Since ML-MCLP remains a relatively unexplored field and presents significant computational challenges due to its NP-hard nature, we proposed the Golden Ball Algorithm (GBA) as a solution approach and compared its performance to that of a Genetic Algorithm (GA). The computational results show that both algorithms are capable of producing high-quality solutions within reasonable computation times. Overall, the GBA generally offers better solution quality, while the GA tends to require less computation time, highlighting a trade-off between solution accuracy and computational efficiency. These results confirm the potential of population-based metaheuristics for solving the ML-MCLP and provide a useful foundation for future research. Future work could focus on larger and more complex instances, as well as on the development and comparison of other metaheuristic and hybrid approaches to further improve solution quality and computational efficiency.

References

[1] K. Abbal, Y. Benadada, and M. El Amrani, "Bi-level multi-capacitated facility location problem," in 2020 5th International Conference on Logistics Operations Management (GOL), Rabat, Morocco, 2020, pp. 1–5. doi: 10.1109/GOL49479.2020.9314710.

[2] K. Abbal, M. El Amrani, and Y. Benadada, "Multi-level multi-capacitated facility location problem: Formulation and resolution," in 2022 IEEE 6th International Conference on Logistics Operations Management (GOL), Strasbourg, France, 2022, pp. 1–6. doi: 10.1109/GOL53975.2022.9820242.

[3] M. El-Amrani, Y. Benadada, and B. Gendron, "Generalization of capacitated p-median location problem: Modeling and resolution," in 2016 3rd International Conference on Logistics Operations Management (GOL), Fez, Morocco, 2016, pp. 1–6. doi: 10.1109/GOL.2016.7731674.

[4] D. Tcha and B. Lee, "A branch-and-bound algorithm for the multi-level uncapacitated facility location problem," European Journal of Operational Research, vol. 18, no. 1, pp. 35–43, 1984. doi: 10.1016/0377-2217(84)90258-3.

[5] C. Ortiz-Astorquiza, I. Contreras, and G. Laporte, "An exact algorithm for multilevel uncapacitated facility location," Transportation Science, vol. 53, no. 4, pp. 1085–1106, 2019. doi: 10.1287/trsc.2018.0868.

[6] P. Chardaire, J.-L. Lutton, and A. Sutter, "Upper and lower bounds for the two-level simple plant location problem," Annals of Operations Research, vol. 86, pp. 117–140, 1999. doi: 10.1023/A:1018942415824.

[7] S. Tragantalerngsak, J. Holt, and M. Rönnqvist, "Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem," European Journal of Operational Research, vol. 102, no. 3, pp. 611–625, 1997. doi: 10.1016/S0377-2217(96)00227-5.

[8] Y. Hinojosa, J. Puerto, and F. R. Fernández, "A multiperiod two-echelon multicommodity capacitated plant location problem," European Journal of Operational Research, vol. 123, no. 2, pp. 271–291, 2000. doi: 10.1016/S0377-2217(99)00256-8.

[9] E. Osaba, F. Diaz, and E. Onieva, "Golden ball: A novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts," Applied Intelligence, vol. 41, no. 1, pp. 145–166, 2014. doi: 10.1007/s10489-013-0512-y.

[10] E. Osaba, F. Diaz, R. Carballedo, E. Onieva, and A. Perallos, "Focusing on the Golden Ball metaheuristic: An extended study on a wider set of problems," The Scientific World Journal, vol. 2014, Article ID 563259, 2014. doi: 10.1155/2014/563259.

[11] K. Ruttanateerawichien, W. Kurutach, and T. Pichpibul, "An improved Golden Ball algorithm for the capacitated vehicle routing problem," in Bio-Inspired Computing: Theories and Applications,

Communications in Computer and Information Science, vol. 472, L. Pan, G. Păun, M. J. Pérez-Jiménez, and T. Song, Eds. Berlin, Germany: Springer, 2014, pp. 341–356. doi: 10.1007/978-3-662-45049-9_56.

[12] A. Berahhou and Y. Benadada, "Dynamic vehicle routing problem with simultaneous delivery and pickup, overtime and inventory restriction: Golden Ball algorithm," in 2022 IEEE 6th International Conference on Logistics Operations Management (GOL), Strasbourg, France, 2022, pp. 1–7. doi: 10.1109/GOL53975.2022.9820268.

[13] J. H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence". Ann Arbor, MI, USA: University of Michigan Press, 1975.

[14] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," Multimedia Tools and Applications, vol. 80, no. 5, pp. 8091–8126, 2021. doi: 10.1007/s11042-020-10139-6.

[15] Y.-Y. Leu, L. A. Matheson, and L. P. Rees, "Sequencing mixed-model assembly lines with genetic algorithms," Computers & Industrial Engineering, vol. 30, no. 4, pp. 1027–1036, 1996. doi: 10.1016/0360-8352(96)00050-2.

[16] M. Watanabe, K. Ida, and M. Gen, "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem," Computers & Industrial Engineering, vol. 48, no. 4, pp. 743–752, 2005. doi: 10.1016/j.cie.2004.12.008.

[17] A. Berahhou, Y. Benadada, and K. Bouanane, "Memetic algorithm for the dynamic vehicle routing problem with simultaneous delivery and pickup," International Journal of Industrial Engineering Computations, vol. 13, no. 4, pp. 587–600, 2022. doi: 10.5267/j.ijiec.2022.6.001.

[18] D. R. M. Fernandes, C. Rocha, D. Aloise, G. M. Ribeiro, E. M. Santos, and A. Silva, "A simple and effective genetic algorithm for the two-stage capacitated facility location problem," Computers & Industrial Engineering, vol. 75, pp. 200–208, 2014. doi: 10.1016/j.cie.2014.05.023.

[19] J. Alcaraz, M. Landete, and J. F. Monge, "Design and analysis of hybrid metaheuristics for the reliability p-median problem," European Journal of Operational Research, vol. 222, no. 1, pp. 54–64, 2012. doi: 10.1016/j.ejor.2012.04.016.

[20] A. Rahmani and S. A. MirHassani, "A hybrid Firefly-Genetic algorithm for the capacitated facility location problem," Information Sciences, vol. 283, pp. 70–78, 2014. doi: 10.1016/j.ins.2014.06.002.

[21] M. Besbes, M. Zolghadri, R. Costa Affonso, F. Masmoudi, and M. Haddar, "A methodology for solving facility layout problem considering barriers: Genetic algorithm coupled with A* search," Journal of Intelligent Manufacturing, vol. 31, no. 3, pp. 615–640, 2020. doi: 10.1007/s10845-019-01468-x.

[22] M. M. Nasiri, V. Mahmoodian, A. Rahbari, and S. Farahmand, "A modified genetic algorithm for the capacitated competitive facility location problem with the partial demand satisfaction," Computers & Industrial Engineering, vol. 124, pp. 435–448, 2018. doi: 10.1016/j.cie.2018.07.045.

[23] J. H. Jaramillo, J. Bhadury, and R. Batta, "On the use of genetic algorithms to solve location problems," Computers & Operations Research, vol. 29, no. 6, pp. 761–779, 2002. doi: 10.1016/S0305-0548(01)00021-1.

[24] Z. Stanimirović, J. Kratica, and D. Dugošija, "Genetic algorithms for solving the discrete ordered median problem," European Journal of Operational Research, vol. 182, no. 3, pp. 983–1001, 2007. doi: 10.1016/j.ejor.2006.09.069.

[25] G. Cornuéjols, R. Sridharan, and J. M. Thizy, "A comparison of heuristics and relaxations for the capacitated plant location problem," European Journal of Operational Research, vol. 50, no. 3, pp. 280–297, 1991. doi: 10.1016/0377-2217(91)90261-S.

[26] A. Klose, "An LP-based heuristic for two-stage capacitated facility location problems," Journal of the Operational Research Society, vol. 50, no. 2, pp. 157–166, 1999. doi: 10.1057/palgrave.jors.2600675.

[27] A. Azadeh, S. Elahi, M. H. Farahani, and B. Nasirian, "A genetic algorithm-Taguchi based approach to inventory routing problem of a single perishable product with transshipment," Computers & Industrial Engineering, vol. 104, pp. 124–133, 2017. doi: 10.1016/j.cie.2016.12.019.