Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 954249, 14 pages
doi:10.1155/2012/954249

Research Article

A Hybrid Approach Using an Artificial Bee Algorithm with Mixed Integer Programming Applied to a Large-Scale Capacitated Facility Location Problem

Guillermo Cabrera G.,[1,2] Enrique Cabrera,[3,4] Ricardo Soto,[1,5] L. Jose Miguel Rubio,[1,6] Broderick Crawford,[1] and Fernando Paredes[7]

[1] *Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile*
[2] *Department of Engineering Science, University of Auckland, Auckland 1020, New Zealand*
[3] *Instituto de Estadística, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile*
[4] *CIMFAV Facultad de Ingeniería, Universidad de Valparaíso, Valparaíso 2362735, Chile*
[5] *Universidad Autónoma de Chile, Santiago 7500138, Chile*
[6] *Departamento de Computación e Informática, Universidad de Playa Ancha, Valparaíso 33449, Chile*
[7] *Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago 8370109, Chile*

Correspondence should be addressed to Guillermo Cabrera G., guillermo.cabrera@ucv.cl

Received 6 September 2012; Revised 12 November 2012; Accepted 14 November 2012

Academic Editor: Rui Mu

We present a hybridization of two different approaches applied to the well-known Capacitated Facility Location Problem (CFLP). The Artificial Bee algorithm (BA) is used to select a promising subset of locations (warehouses) which are solely included in the Mixed Integer Programming (MIP) model. Next, the algorithm solves the subproblem by considering the entire set of customers. The hybrid implementation allows us to bypass certain inherited weaknesses of each algorithm, which means that we are able to find an optimal solution in an acceptable computational time. In this paper we demonstrate that BA can be significantly improved by use of the MIP algorithm. At the same time, our hybrid implementation allows the MIP algorithm to reach the optimal solution in a considerably shorter time than is needed to solve the model using the entire dataset directly within the model. Our hybrid approach outperforms the results obtained by each technique separately. It is able to find the optimal solution in a shorter time than each technique on its own, and the results are highly competitive with the state-of-the-art in large-scale optimization. Furthermore, according to our results, combining the BA with a mathematical programming approach appears to be an interesting research area in combinatorial optimization.

1. Introduction

Heuristics and bioinspired techniques have become efficient and effective alternatives for researchers in solving several complex optimization problems. These types of techniques are able to provide satisfactory solutions for most of the applied problems within acceptable computational times. However, in spite of their effectiveness, these techniques are not able to reach the optimal solution (or ensure its optimality) for large-scale combinatorial optimization problems. In contrast, mathematical programming techniques, particularly the Mixed Integer Programming (MIP), have been studied and developed by scholars over several decades with the main goal of obtaining optimal solutions to difficult problems using as little CPU time as possible. In this case, researchers must face the tradeoff between computational time and the quality of the result. For these reasons, the combination of metaheuristics and various mathematical approaches has become a well-studied area. Interested readers can find two recent and comprehensive works on the hybridization of stochastic techniques and mathematical programming (MP) approaches in [1, 2].

Swarm Intelligence (SI), as well as other mathematical programming techniques, has been applied successfully to several difficult combinatorial optimization problems (see [3–6] for a range of applications). Additionally, as mentioned above, hybrid strategies have been developed to improve the effectiveness of these techniques. A more complete review is provided in Section 2.

The Artificial Bee algorithm (BA) is a relatively new approach in SI. Originally proposed in 2006 [7], it was mainly inspired by the foraging behavior of honeybees and has been applied to a range of problems in both combinatorial and functional optimizations with highly successful results. The BA has also been applied to large-scale problems [8] and has outperformed other well-known swarm-based algorithms, including Particle Swarm Optimization (PSO) and Differential Evolution (DE). However, as with almost all stochastic approaches, the BA cannot ensure optimality even when the optimal solution is found. For small- and medium-size instances, this limitation is not highly relevant because heuristics techniques have empirically demonstrated their ability to achieve convergence in quite acceptable time; therefore, solutions provided by stochastic approaches are likely to be optimal (or a very good approximation) in large-scale instances. However, when large-scale optimization is considered, it is not possible to know how "good" the solution provided by the heuristic.

In this paper, we propose a hybrid algorithm of the BA and the MIP for solution of several large-scale instances of the well-known Capacitated Facility Location Problem (CFLP). The CFLP is one of the most important problems for companies that distribute products to their customers. The problem consists of selecting specific sites at which to install plants, warehouses, and distribution centers while assigning customers to service facilities and interconnecting facilities using flow assignment decisions. This paper considers a two-level supply chain in which a single plant serves a set of warehouses, which in turn serve a set of end customers or retailers. Figure 1 shows the basic configuration of our supply chain. Therefore, we aim to solve this problem by finding a set of locations that allow us to serve the entire set of customers in an optimal way. As Figure 1 shows, each customer (or cluster) is served only by one warehouse.

Despite its good performance in several optimization problems, the BA is not able to provide an optimal solution for large-scale problems. Furthermore, it is possible to become trapped in a local optimum. To bypass these drawbacks, we propose a hybrid algorithm that selects a subset of promising centers using the BA and subsequently solves the subproblem

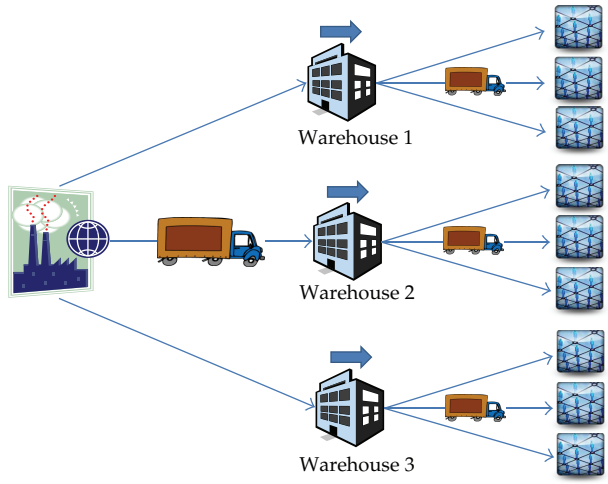

Figure 1: A two-level supply chain network configuration.

using a simple MIP algorithm. The BA guides the process while the MIP provides the optimal values of the simplified problems. One distinctive feature of our algorithm is that it solves the primary problem directly using the MIP algorithm, which is possible due to the reduction of the search space produced by the BA algorithm.

Although several works that have proposed various hybrid approaches to solve the CFLP and its uncapacitated (UFLP) version exist in the literature (e.g., [9, 10]), we are unaware of any prior publications that use the Bees Algorithm (BA) to solve a large-scale CFLP. Moreover, we have found no articles that hybridize the BA with an MP approach in any optimization problem. Therefore, one contribution of this paper is the presentation of a performance analysis for the hybrid BA-MIP algorithm. A second contribution is the application of the BA to a large-scale problem that provides optimal solutions rather than only locally optimal solutions.

The remainder of this paper is organized as follows. Section 2 presents an overview of the CFLP and BA concepts. The hybrid BA-MIP algorithm is covered in Section 3, and a detailed explanation of the algorithm is also presented in this section. Section 4 begins with a brief description of the benchmarks used in this paper, and the experimental results are subsequently presented and discussed. Finally, Section 5 outlines selected conclusions.

2. Literature Review

This section presents a literature review. Section 2.1 discusses the mathematical model for CFLP and provides relevant background for certain approaches presented in the literature. Section 2.2 provides an overview of the BA and highlights its main features.

2.1. Capacitated Facility Location Problem

The CFLP in this work contains a set of warehouses that supply a set of customers that are uniformly distributed in a limited area. The model considers the installation cost (i.e., the cost associated with opening a specific warehouse) and transportation or assignation cost

(i.e., the cost related to transportation of a specific amount of products from a warehouse to a customer). The mathematical model for the CFLP is presented as follows:

$$\sum_{i=1}^{N} F_i X_i + \sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij} Y_{ij}, \tag{2.1}$$

s.t.:

$$\sum_{i=1}^{N} Y_{ij} = 1, \quad \forall j = 1, \ldots, M \tag{2.2}$$

$$Y_{ij} \leq X_i, \quad \forall i = 1, \ldots, N, \quad \forall j = 1, \ldots, M \tag{2.3}$$

$$\sum_{j=1}^{M} \mu_j Y_{ij} \leq I_i^{\mathrm{CAP}}, \quad \forall i = 1, \ldots, N \tag{2.4}$$

$$X_i \in \{0, 1\}, Y_{ij}[0, 1], \quad \forall i = 1, \ldots, N; \quad \forall j = 1, \ldots, M. \tag{2.5}$$

Equation (2.1) represents the total system cost. The first term denotes the fixed setup and operating cost for opening warehouses, and the second term indicates the daily transport cost between the warehouse and the customers. Equation (2.2) ensures that the customer demands are completely served by the system. Equation (2.3) ensures that the customers are assigned to the installed warehouses ($X_i = 1$). Equation (2.4) states that the summation of the demand μ of each customer served by a particular warehouse i must be less than or equal to a threshold I_i^{CAP}, which can be different for each warehouse. Finally, (2.5) states the integrality (0-1) for the variable X_i and sets the range of the variable Y_{ij}. This model is NP-hard because it is clearly an extension of the UFLP, which is known to be NP-hard. Additionally, this model is one of the most basic examples related to location research, and several comprehensive surveys on location theory can be found in [11, 12].

The CFLP is well known in the operational research literature, and several authors have tackled this problem using different techniques (e.g., Genetic Algorithms are implemented in [13], and the Tabu Search (TS) algorithm is used in [14]). A comparison of the performance of these heuristics is provided in [15]. Additionally, mathematical-based approaches have been extensively developed to solve the CFLP, and algorithms based on Lagrangian relaxation represent the most common math-based approach [4, 8, 16]. In [17], the authors develop a column generation strategy to obtain the exact solution for large-scale instances. Other mathematical approaches are revised in [18, 19]. Moreover, mixed approaches using MP techniques and heuristics have been previously proposed. For instance, in [20], the authors develop a Lagrangian-based heuristic (LH) that provides lower bounds to the problem, and the TS algorithm is subsequently used to find the upper bound of the problem. In this case, the TS is initialized using the primary information provided by LH. Additionally, in [21], the authors combine Lagrangian relaxation with Ant Colony Optimization (ACO). Although the CFLP is one of the most studied models in combinatorial optimization, to the best of our knowledge, no BA study has tackled this problem.

2.2. Bees Algorithm

The Bees Algorithm (BA) is a nature-inspired approach that was originally proposed by Pham et al. [7] to solve complex optimization problems. Subsequently, several authors have applied different versions of the BA to tackle a wide range of combinatorial and continuous optimization problems. The algorithm has been demonstrated to be highly competitive, especially when compared with other swarm/population-based approaches such as the PSO [22] or Genetic Algorithms [23]. Karaboga and Akay [24] provide a comprehensive literature review and a comparison of the most important swarm/population-based approaches. A complete survey of various BA applications is also provided by Karaboga and Akay in [25]. In the following section, we present a brief description of the general structures of our BA. This description is mainly based on the descriptions provided by [7, 24–26].

One of the most important characteristics of swarm intelligence is the ability to exchange relevant information among individuals. This feature allows the swarm to generate and develop collective knowledge. In the case of the BA, this information addresses the recognition of promising sources of food found by any individual insect of the swarm. Another relevant feature of the bee swarm is the ability to intensify the search in certain patches that have been identified as promising. The two main attributes of most heuristics used in optimization are exploration and exploitation. The first provides a fast and wide search throughout the search space (which is usually too large). The second allows an intensive search of certain reduced search spaces (neighborhoods) that have been identified as "good-quality patches" during the exploration phase. In the BA case, the scout bees provide the exploration characteristics. The scout bees seek the search space in a (usually) random manner. Each scout bee visits a patch and evaluates it, and the scout bees have the ability to communicate the quality of the patch (fitness) to the unemployed bees. Depending on the attractiveness of each patch, the unemployed foragers will follow the scout bees to exploit the most promising patches. Once the patch is no longer attractive, a subset of bees will continue to search while the others wait for another promising patch. As in other heuristics, the balance between exploitation and exploration is a notably important issue for the BA. If we prioritize the exploration phase of the BA, it is likely to suffer from rather slow convergence. However, if we prioritize the exploitation phase of the BA, it is likely to become trapped in local minima. In most of the swarm optimization algorithms as well as other heuristics (e.g., Tabu Search), the BA uses memory structures to influence the next population (swarm). In this case, two main strategies provide information from former population to the new population. The first strategy uses experienced foragers, that is, bees with notably good fitness are included in the next population. The second strategy includes the best bee from a subset of the high-quality patches. In this manner, the use of various elements will determine the balance between exploration and exploitation.

3. Hybrid BA and LP Algorithm

In this paper, we present a BA that is hybridized with a MIP algorithm provided by GUROBI. Our algorithm contains the following distinctive features.

3.1. Variable Neighborhood Sizes

We use three different neighborhood sizes. It is important to note that we only refer to the "*size*" and not the "*structure*" of the neighborhood. More specifically, the neighborhood

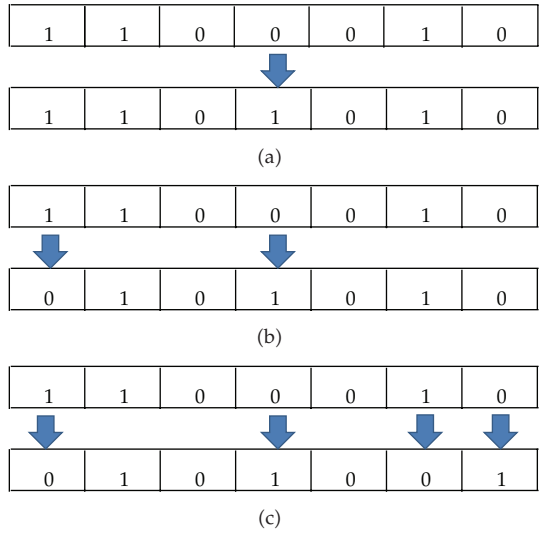

Figure 2: Three implemented neighborhood sizes: (a) *small-size* movement, (b) *medium-size* movement, and (c) *large-size* movement.

structure does not change during the algorithm execution. Figure 2 shows the three sizes and their function in the experiments. Figure 2(a) shows a *small-size* move in which only one location is modified (in this case, warehouse 4 is opened), and Figure 2(b) presents an example of a medium-size move. In this case, two locations are modified (warehouse 1 is closed, and warehouse 4 is opened). Figure 2(c) shows a large-size move.

Using notation from [27], we describe our variable neighborhoods as follow: let S denote any subset of open warehouses ($S \subseteq M$); the solution space $\Im$ may be subsequently defined as all such possible subsets. The total number of solutions in $\Im$ is $2^m - 1$. To define the neighborhoods, [27] defines a distance function as follows: let S_1, S_2 be any two solutions in $\Im$; the distance between them is defined by $\rho(S_1, S_2) = |(S_1 \setminus S_2) \cup (S_1 \setminus S_2)|$. Therefore, the distance between one solution and another will increase when allocation of a specific customer in S_1 is different from the allocation of the same customer made in S_2. An intuitive consequence of the above is that the distance between solutions S_1, S_2 is zero if and only if $S_1 = S_2$. Figure 1 shows an example of the distance between the different solutions. For instance, in Figure 2(a), $\rho(S_1, S_2) = 1$; in Figure 2(b), $\rho(S_1, S_2) = 2$; and in Figure 2(c), $\rho(S_1, S_2) = 4$. The values of $\rho()$ used in our algorithm are shown at the end of Section 3. It is important to note that we do not implement a variable neighborhood strategy as in [27]. Instead, we define different neighborhood structures that are only applied for quite specific tasks during the execution of our algorithm.

(i) *Neighborhood-based start strategy*: the set of scout bees is initialized with a neighborhood-based strategy that uses the "large" movement to move from one solution to the other. This strategy allows us to use the LP solver more efficiently without compromising the algorithm performance.

(ii) *Intensification procedure:* we implement an intensification procedure to exploit the promising patches found by the bees. When a promising solution is found, the intensification procedure is triggered, and a local search using a rather "small" neighborhood movement is carried out. The intent behind this procedure is to find

a local (and hopefully global) minimum as fast as possible, and this minimum is likely to be located near the promising solutions.

(iii) *Roulette-wheel selection procedure*: we implement a procedure based on the well-known roulette wheel to select the elite bees and to assign the follower bees to particular elite (or selected nonelite) bees. This procedure allows for the inclusion of selected diversification mechanisms during the algorithm execution.

(iv) *Use of experienced foragers:* we include the information of the best solution in the form of "*experienced foragers*" in each iteration of the algorithm, which allows us to include certain historical information to improve the convergence of the algorithm.

The algorithm begins with a set of *ns* scout bees, which are initialized using the neighborhood-based start strategy. Other techniques have found that certain "warm" start solutions could be implemented as well. However, in our practical experience, our approach is an easier and faster method of initializing the set of scout bees mainly due to the efficiency mechanisms of the solver. Once the *ns* scout bees are generated, they are sorted according to their fitness. The fitness is calculated using the objective function (2.1), which corresponds to an attractiveness measure in relation to the other scout bees. Note that the cost of each bee corresponds to the optimal solution for the subset of warehouses. This optimal value is provided by the MIP solver in less than two seconds on average. Therefore, this approach allows us to carry out a "global" optimization in different subspaces of the problem. Equations (3.1) and (3.2) state this attractiveness measure:

$$\sum_{b \in S} f(b) = tc, \tag{3.1}$$

$$\text{fitness}_b = \left(\frac{f(b)}{tc} \right), \tag{3.2}$$

where S is the set of scout bees and $f(x)$ is our objective function (2.1). The total cost of the set of scout bees is calculated n (3.1) and in (3.2), and the fitness is calculated based on the fitness of a specific bee and the total cost of the swarm. Because we are aiming to optimize the fitness, our results must be normalized such that the lowest cost becomes the most attractive. Once the scout bees have been sorted, the e bees (elite bees) are selected using the roulette-wheel selection procedure. We note that, if there are important differences among the costs of the *ns* scout bees, then the algorithm will likely select the e most attractive ones. In the same way, the *ne* (nonelite) bees are selected from the remaining scout bees. Subsets e and *ne* are sorted, and the attractiveness of each bee is calculated as explained above. Next, a set F (followers) is generated and assigned to the both elite and nonelite bees. Because the roulette-wheel selection procedure is used again, the elite bees are likely to recruit more followers than the less attractive nonelite bees. A local search is carried out at this stage using medium-size movements. Once the local search is completed, the algorithm selects the best bee from each patch. If a new best solution is reached, the intensification procedure is triggered. Following that procedure, the entire swarm is sorted based on fitness. Next, the elite and nonelite classification is executed again. In this step, a few random bees are added to the swarm to diversify the search. Additionally, the best solution is included via an experienced forager. The pseudocode for our hybrid BA-MIP algorithm is presented as follows Algorithm 1.

(1) **Init**
(2) Generate initial swarm S
(3) Solve MIP of each Sb
(4) Sort S based on fitness
(2.5) Select elite and non-elite bees
(3.1) **While** not end-criterion
(3.2) Assign followers
(8) **For each** elite and non-elite
(9) Search η(E) $\cup$ η(NE)
(10) **End For**
(11) Select Best Bee (bBEST)
(12) **If** bBEST < BestSol
(13) BestSol $\leftarrow$ bBEST
(14) Intensification
(15) **End If**
(16) Joint and Sort F , E , NE
(17) Sort S based on fitness
(18) Select elite and non-elite bees
(19) Add Random Bees
(20) Add BestSol Bee
(21) **End While**
(22) **End**

Algorithm 1: Hybrid BA-MIP algorithm.

Table 1: Results of the parameter tuning process.

Item	Test	Best
ns	10; 20	**10**
e	10%; 20%; 50% of *ns*	**20**%
ne	10%; 40% of *ns*	**10**%
Followers	20; 100	**20**
Neigh small	1; 2	**2**
Neigh med	4; 6	**4**
Intensification size	10; 20	**20**
Elite followers	60%; 90%	**60**%

3.2. Parameter Tuning

To obtain a parameter set for the BA algorithm, we performed several tests using different values for each parameter of the algorithm, and these values are presented in Table 1. The tests were applied on one of the medium-size instances. The selected values are shown in bold.

4. Computational Experiments and Discussion

This section explains the experiments, certain benchmarks from the literature included to validate our algorithm, and the generation of a set of large-scale instances. The results obtained from our BA-MIP algorithm are compared with those from a simple local search

Table 2: Optimal DND obtained for both the ILM-PR and ILM-CR models.

Instance	Opt	BA	BA + MIP	LS + MIP			
		Time	GAP	Time (sec)	GAP	Time (sec)	GAP
capa	19,240,822.449	Max	>300%	426.022	0%	367.735	0%
capb	13,656,379.578	Max	>300%	345.501	0%	317.468	0%
capc	11,646,596.974	Max	>300%	553.624	0%	129.336	0%

strategy (which uses the MIP to solve the allocation problem) as well as those from both the MIP algorithm and the BA applied independently. A comprehensive analysis and discussion is outlined at the end of this section.

4.1. Experiments and Computational Results

In this subsection, we present the benchmarks applied for performance comparison and the computational results obtained for the hybrid algorithm. Finally, we show a summary of the principal results obtained. The computational experiments were performed on an Intel Core Duo processor CPU T2700, 2.33 GHz with 2 GB of RAM and Windows XP operating system. The BA algorithm was implemented in the JAVA programming language using NetBeans IDE, and the MIP algorithm was modeled with the GUROBI solver.

To validate the algorithm and measure its convergence, we chose a set of medium-size instances that have known optimal solutions, namely, capa, capb, and capc. These instances were obtained from Beasley's ORLibrary (http://people.brunel.ac.uk/~mastjjb/jeb/info.html). Additionally, a set of large instances (300 warehouses and 1000 clients, 500 warehouses and 1000 clients, and 1000 warehouses and 1000 clients) was created using the strategy provided in [8]. The set of customers and the set of warehouses are uniformly distributed over a plane of 10×10 distance units. The Euclidean distance between a customer i and a warehouse j corresponds to the transportation cost Y_{ij}. The demand d_j is calculated using a uniform distribution U [5, 35]. The I_i^{CAP} is calculated using U [100, 1600], and we amplify the capacity of the warehouse to obtain harder instances. Finally, the fixed cost of warehouse i is calculated by $F_i = U[0, 90] + U[100, 110]\sqrt{I_i^{\mathrm{cap}}/10}$. This expression takes into account the economies of scale [8]. As proposed in [8], we generate three different classes of problems: 300×1000, 500×1000, and 1000×1000 (warehouses × customers). To avoid any instance-dependent effects, we generated 10 different instances for each class. We also executed the algorithm 30 times for each instance to assess and avoid outlier performance. The results presented in this section correspond to the average values from these experiments. The MIP algorithm was executed only once per instance due to its deterministic behavior, and the results obtained from the MIP algorithm are presented in Table 2. The stop criterion used in the three first instances (capa, capb, and capc) was GAP $\approx 0\%$, that is, we forced the algorithm to find the (known) optimal solution. Because the optimal solution value is not known for our generated large-scale instances, the algorithm was aborted after 2000 seconds or GAP $\leq 1\%$, whichever occurred first.

Figures 3 and 4 show the convergence of two algorithms (MIP and BA + MIP) for the instances 500_3 and 1000_4, respectively.

As shown in Figure 3 for the 500_3 instance, the two algorithms are both able to find notably good solutions within the time allotted. As expected, the MIP requires more CPU time than the BA-MIP hybrid algorithm to find the solution, and the solid line shows the LB

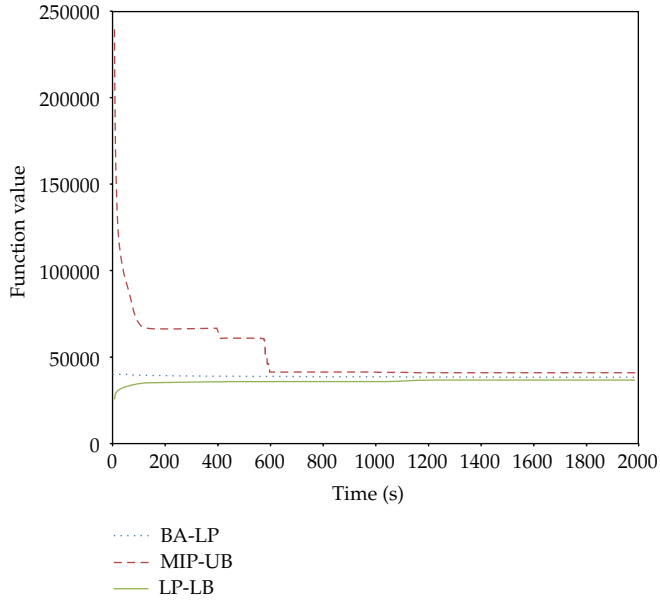

Figure 3: Comparison of the convergence between the MIP solver and the BA-MIP algorithm (instance 500_3).

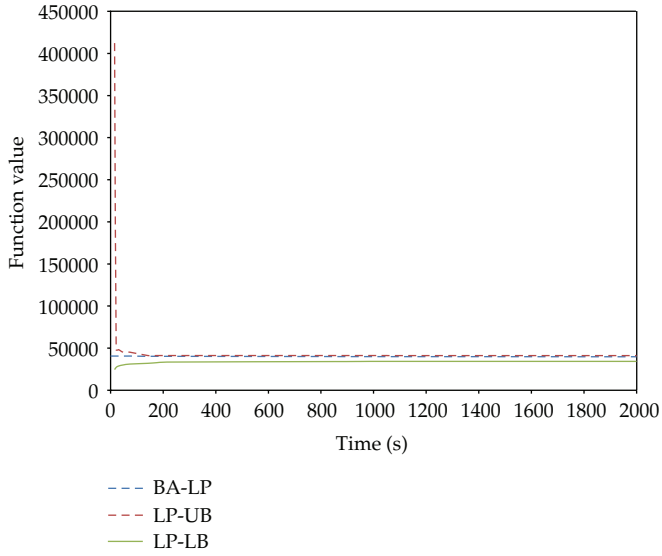

Figure 4: Comparison of the convergence between the MIP solver and the BA-MIP algorithm (instance 1000_4).

of this instance. A remarkable feature of our hybrid approach is its "warm" initial solution. As shown in Figure 3, the first iterations produce very good solutions that are quite close to the best solution reached by the MIP algorithm after the 2,000 seconds. We assume that this observation is due to our *neighborhood-based start strategy*.

Figure 4 shows a similar situation for the 1000_4 instance. The main difference in this case is the rapid convergence of the MIP algorithm. Despite this observation, the MIP is not able to find a better solution than that obtained by our BA-MIP algorithm.

Table 2 shows the average results obtained by the four algorithms for the three medium-size instances. The *Opt* column shows the optimal value produced for the respective instance. The *Time* column reports the CPU time required by the respective algorithm to reach the optimal value. The GAP column shows the difference between the average value and the optimal value as a percentage. A value of GAP = 0% indicates that the optimal value was reached, and GAP > 0% otherwise. The maximum time available for each instance corresponds to the time in which the MIP algorithm was able to find the optimal value for that instance.

As shown in Table 2, the BA + MIP and the LS + MIP were able to find the optimal solution more quickly than the MIP solver.

These results demonstrate that the robustness of our algorithm highlights the speed of our hybrid approach, which could be a determining factor for notably large-scale instances in which the MIP is not able to find the optimal value within a reasonable CPU time.

The *Time* column in Table 3 shows the time (in seconds) in which the best value was found by the respective algorithm. The GAP column shows the difference between the average value and the lower bound value as a percentage. As stated previously, the maximum time available for each instance was 2000 seconds. The excellent performance of our hybrid approach is quite obvious; it outperforms the MIP approach in almost all cases, especially those instances in which the number of warehouses is greater than or equal to 500. This situation demonstrates the robustness of our algorithm as well as its reliability. It is also important to note that, in most of the cases, the best value obtained by the MIP algorithm was improved upon by our BA-MIP algorithm before 1500 seconds had elapsed, which implies a time saving of 25% compared with the mathematical approach. Moreover, in most cases, the best value provided by our hybrid algorithm was found near the time limit, which could be considered as a quite promising feature because it means that our algorithm is able to escape from local optima. Additionally, we note that, in the few cases in which the MIP reached a better value than our hybrid approach did, the difference is not greater than 1% of the GAP. The results obtained from the simple BA algorithm are surprisingly poor. We believe this situation may be partially due to the size of the search space as well as to a lack of precision in the parameter tuning. However, even when a careful parameter tuning process was conducted, the improvement is only marginal if we consider the current GAP between the LB for each instance and the UB obtained by BA. However, the LS-MIP obtains very good results for all instances. These results confirm that even rather simple heuristic strategies can be significantly improved when combined with mathematical programming approaches. However, we note that most of the best values were obtained early in the time elapsed, which may mean that the LS-MIP algorithm is not able to explore large spaces and quickly converges to local minima. Despite this weakness, the good performance shown by the LS-MIP algorithm encourages further exploration with different hybrid approaches.

Table 4 summarizes the results (GAP, as a percentage) obtained by our three algorithms (the BA is not considered) independent of any instance-dependent effects. It can be clearly observed that despite the effects provoked by particular instances, our algorithm presents the most desirable features, that is, a reduced mean, which suggests that the result is closer to the LB on average, and the small variance can be interpreted as a measure of robustness and reliability.

Table 3: Obtained results per algorithm for all (very) large instances.

Instance	GAP MIP	BA	BA + MIP	LS + MIP			
		Time (sec)	GAP (%)	Time (sec)	GAP (%)	Time (sec)	GAP (%)
300_1	2.54	1,083.566	319.94	1,191.68	**2.52**	823,748	3.48
300_2	4.32	521.891	326.39	1,114.84	**1.22**	1,859.55	1.23
300_3	5.38	418.646	344.74	1,183.13	5.67	67.170	**5.47**
300_4	**0.25**	576.624	325.64	1,354.32	0.53	291.096	0.92
300_5	7.69	499.160	340.61	1,181.11	**3.60**	248,047	5.03
300_6	**0.10**	0.440	311.72	1,779.96	0.76	1,068.721	0.87
300_7	7.24	1,532.097	333.29	1,942.37	**4.16**	810.303	4.82
300_8	3.39	247.004	322.63	1,716.07	**0.91**	122.122	2.13
300_9	6.24	1,401.028	331.60	842.08	**4.55**	446.381	4.89
300_10	3.82	1,786.086	331.14	1111016	**2.14**	82.730	4.68
500_1	4.30	1,515.316	425.51	1,382.22	4.41	656.431	**3.43**
500_2	8.34	29.623	432.50	1,960.32	**4.43**	400.010	5.73
500_3	11.61	741.206	428.80	1,636.51	**4.51**	83.878	7.23
500_4	8.54	1,740.442	441.34	1,476.66	4.62	519.690	**3.78**
500_5	7.27	364.557	437.55	1,742.31	**4.83**	816.807	4.98
500_6	8.09	151.569	430.50	1,306.528	**5.20**	328.605	6.00
500_7	6.63	1,493.073	425.32	1,892.530	**4.48**	1,093.399	5.07
500_8	11.18	1,357.436	420.11	1,177.792	4.98	828,960	**4.14**
500_9	9.64	853.731	427.62	1,647.911	5.16	860.271	**4.69**
500_10	12.88	1,826.382	423.29	1,333.395	**5.99**	231,932	6.64
1000_1	18.75	89.000	331.06	1,514.020	**15.62**	246.482	17.13
1000_2	22.97	92.000	336.27	1,641.200	15.65	1,210.875	**14.30**
1000_3	20.33	85.000	337.88	1,979.140	16.45	1,165.605	**15.22**
1000_4	20.06	1,842.591	339.48	1,954.162	**16.10**	1,997.213	17.97
1000_5	17.91	88.000	332.41	1,974.203	16.64	305,483	17.60
1000_6	16.84	90.000	340.57	1,993.361	14.72	741838	**14.41**
1000_7	18.80	88.000	339.95	1,898.845	**17.30**	104.692	17.80
1000_8	19.20	89.000	322.16	1,948.752	**14.31**	1,206.733	14.76
1000_9	17.62	1,807.162	340.03	1,737.584	15.87	378.913	**14.99**
1000_10	18.93	317.728	340.53	1,074.975	17.25	609,155	**16.98**

Table 4: Summary of the results obtained by the MIP, LS-MIP, and BA-MIP.

	MIP	LS-MIP		BA-MIP		
Instances						
300x	4.14	6.47	3.35	3.14	**2.61**	**2.93**
500x	8.85	5.89	5.17	1.37	**4.86**	**0.22**
1000x	19.14	2.64	16.12	2.04	**15.99**	**0.87**

5. Conclusions and Future Work

In this paper, we have presented a hybrid algorithm based on the BA and the MIP and used it to solve the CFLP. The BA is applied primarily for the purpose of solving the location problem, that is, finding a subset from the available set of locations that satisfy the entire demand of the system. In contrast, the MIP is applied for the purpose of finding the optimal

solution by considering a specific subset of locations, that is, solving the allocation problem. Certain considerations are important and must be highlighted in this approach.

First, because the algorithm is able to find the optimal solution for each subset of selected warehouses, we are able to fairly compare those subsets. At the same time, our hybrid algorithm is able to find the optimal solution to medium-large problems, which is not possible with the use of common local search strategies. Additionally, obtaining the optimal solution allows us to compare the different strategies for warehouse selection and local search.

Second, compared with the widely used local search approaches that implement a swap in the assignation vector as a neighborhood move, our approach is able to visit more solutions because the entire tree corresponding to the feasible allocations is considered for each subset of warehouses in the search for the optimal solution of the subproblem by the MIP algorithm.

Third, our BA does not require extensive computational resources because most of the calculations are handled by the MIP solver, which is by far more efficient than many common heuristics implementations.

Our hybrid BA-MIP algorithm was able to find the optimal solution for a set of well-known large-scale instances found in the literature. Moreover, it outperformed both the BA and MIP approaches as applied separately. Compared with the state-of-the-art algorithms for location problems, our BA-MIP algorithm is highly competitive and reaches an optimal solution using less CPU time than both exact and stochastic approaches. Moreover, when the instances are notably large, the algorithm is able to find a better solution than that of the MIP approach in a fraction of the time. As widely reported in the literature over the last three decades, the combination of mathematical programming with heuristics and/or metaheuristics (Matheuristics [1]) provides robust and straightforward approaches to solving these types of problems.

In this paper, we have developed a hybrid algorithm using BA and LP using an approach that has not yet been reported. Due to the hybrid nature of the algorithm, certain changes were made in the structure of the BA heuristic. Use of the variable neighborhood "size," the neighborhood-based initialization procedure, the intensification procedure, and the experienced foragers are the most important features of our implementation. These distinctive components allow us to improve the performance of the simple BA when combined with MIP.

The hybridization of BA with other such mathematical programming approaches as interior point methods, column generation, and gradient-based algorithms shows promise as a potentially valuable research area. Additionally, the application of similar hybrid approaches to more complex large-scale problem will be an interesting future research line.

References

[1] V. Maniezzo, T. Stützle, and S. Voß, Eds., *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, vol. 10, Annals of Information Systems, 2009.

[2] C. Blum, J. Puchinger, G. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: a survey," *Applied Soft Computing Journal*, vol. 11, no. 6, pp. 4135–4151, 2011.

[3] G. Cabrera, S. Roncagliolo, J. P. Riquelme, C. Cubillos, and R. Soto, "Hybrid particle swarm optimization—simulated annealing algorithm for the probabilistic traveling salesman problem," *Studies in Informatics and Control (SIC)*, vol. 21, no. 1, pp. 49–58, 2012.

[4] J. E. Beasley, "Lagrangean heuristics for location problems," *European Journal of Operational Research*, vol. 65, pp. 383–399, 1993.

[5] M. H. Kashan, N. Nahavandi, and A. H. Kashan, "DisABC: a new artificial bee colony algorithm for binary optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 342–352, 2012.
[6] C. Canel and B. M. Khumawala, "A mixed-integer programming approach for the international facilities location problem," *International Journal of Operations & Production Management*, vol. 16, no. 4, pp. 49–68, 1996.
[7] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm—a novel tool for complex optimization problems," in *Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS '06)*, pp. 454–459, 2006.
[8] F. Barahona and F. A. Chudak, "Near-optimal solutions to large-scale facility location problems," *Discrete Optimization*, vol. 2, no. 1, pp. 35–50, 2005.
[9] M. Caserta and E. Quiñonez Rico, "A cross entrophy-based metaheuristic algorithm for large-scale capacitated facility location problem," *Journal of the Operational Research Society*, vol. 60, pp. 1439–1448, 2009.
[10] K. S. Al-Sultan and M. A. Al-Fawzan, "A tabu search approach to the uncapacitated facility location problem," *Annals of Operations Research*, vol. 86, pp. 91–103, 1999.
[11] M. S. Daskin, *Network and Discrete Location*, John Wiley & Sons, New York, NY, USA, 1995.
[12] Z. Drezner and H. W. Hamacher, *Facility Location: Applications and Theory*, Springer, New York, NY, USA, 2004.
[13] M. J. Cortinhal and M. E. Captivo, "Genetic algorithms for the single source capacitated location problem," in *Metaheuristics: Computer Decision-Making*, M. G. Resende and J.P. de Sousa, Eds., pp. 187–216, Kluwer Academic Publisher, 2004.
[14] M. H. Sun, "A tabu search heuristic procedure for the capacitated facility location problem," *Journal of Heuristics*, vol. 18, no. 1, pp. 91–118, 2012.
[15] M. A. J. Arostegui, S. N. Kadipasaoglu, and B. M. Khumawala, "An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems," *International Journal of Production Economics*, vol. 103, pp. 742–754, 2006.
[16] G. Cornuejols, R. Sridharan, and J. M. A. Thizy, "comparison of heuristics and relaxation for the capacitated plant location problem," *European Jurnal of Operational Research*, vol. 50, pp. 280–297, 1991.
[17] A. Klose and A. Drexl, "Lower bounds for the capacitated facility location problem based on column generation," *Management Science*, vol. 51, no. 11, pp. 1689–1705, 2005.
[18] P. Sinha, "Observations on some heuristic methods for the capacitated facility location problem," *Opsearch*, vol. 49, no. 1, pp. 86–93, 2012.
[19] J. K. Sankaran, "On solving large instances of the capacitated facility location problem," *European Journal of Operational Research*, vol. 178, no. 3, pp. 663–676, 2007.
[20] M. J. Cortinhal and M. E. Captivo, "Upper and lower bounds for the single source capacitated location problem," *European Journal of Operational Research*, vol. 151, no. 2, pp. 333–351, 2003.
[21] C. Chen and C. Ting, "Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem," *Transportation Research E*, vol. 44, no. 6, pp. 1099–1122, 2008.
[22] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*, Morgan Kaufman, 2001.
[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.
[24] D. Karaboga and B. Akay, "A comparative study of artificial Bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.
[25] D. Karaboga and B. A. Akay, "survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, pp. 61–85, 2009.
[26] L. Özbakir, A. Baykasoğlu, and P. Tapkan, "Bees algorithm for generalized assignment problem," *Applied Mathematics and Computation*, vol. 215, no. 11, pp. 3782–3795, 2010.
[27] P. Hansen, J. Brimberg, D. Urošević, and N. Mladenović, "Primal-dual variable neighborhood search for the simple plant-location problem," *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 552–564, 2007.