Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 1506084, 7 pages
http://dx.doi.org/10.1155/2016/1506084

Research Article

A Matheuristic Approach Combining Local Search and Mathematical Programming

Carolina Lagos,[1] Guillermo Guerrero,[1] Enrique Cabrera,[2] Stefanie Niklander,[3,4] Franklin Johnson,[5] Fernando Paredes,[6] and Jorge Vega[7]

[1] *Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile*
[2] *CIMFAV-Facultad de Ingeniería, Universidad de Valparaíso, 2374631 Valparaíso, Chile*
[3] *Universidad Autónoma de Chile, 7500138 Santiago, Chile*
[4] *Universidad Cientifica del Sur, Lima 18, Peru*
[5] *Universidad de Playa Ancha, Casilla 34-V, Valparaíso, Chile*
[6] *Escuela de Ingeniería Industrial, Universidad Diego Portales, 8370109 Santiago, Chile*
[7] *Departamento de Ingeniería Eléctrica, Universidad de Antofagasta, 1270300 Antofagasta, Chile*

Correspondence should be addressed to Carolina Lagos; carolina.lagos.c@mail.pucv.cl

Received 31 August 2015; Revised 2 December 2015; Accepted 5 January 2016

Academic Editor: Wan Fokkink

A novel matheuristic approach is presented and tested on a well-known optimisation problem, namely, capacitated facility location problem (CFLP). The algorithm combines local search and mathematical programming. While the local search algorithm is used to select a subset of promising facilities, mathematical programming strategies are used to solve the subproblem to optimality. Proposed local search is influenced by instance-specific information such as installation cost and the distance between customers and facilities. The algorithm is tested on large instances of the CFLP, where neither local search nor mathematical programming is able to find good quality solutions within acceptable computational times. Our approach is shown to be a very competitive alternative to solve large-scale instances for the CFLP.

1. Introduction

The capacitated facility location problem (CFLP) is one of the most important problems for companies which distribute products to their customers. The problem consists of selecting specific sites at which to install plants, warehouses, and distribution centres 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 allows 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.

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 allocation 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:

$$\min \quad \sum_{i=1}^{N} (F_i \cdot X_i) + \sum_{i=1}^{N}\sum_{j=1}^{M} \left(C_{ij} \cdot d_j \cdot Y_{ij}\right) \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{M} d_j \cdot Y_i j \le I_i^{\text{cap}} \cdot X_i \quad \forall i = 1, 2, \ldots, N, \tag{2}$$

$$\sum_{i=1}^{N} Y_{ij} = 1 \quad \forall j = 1, 2, \ldots, M, \tag{3}$$

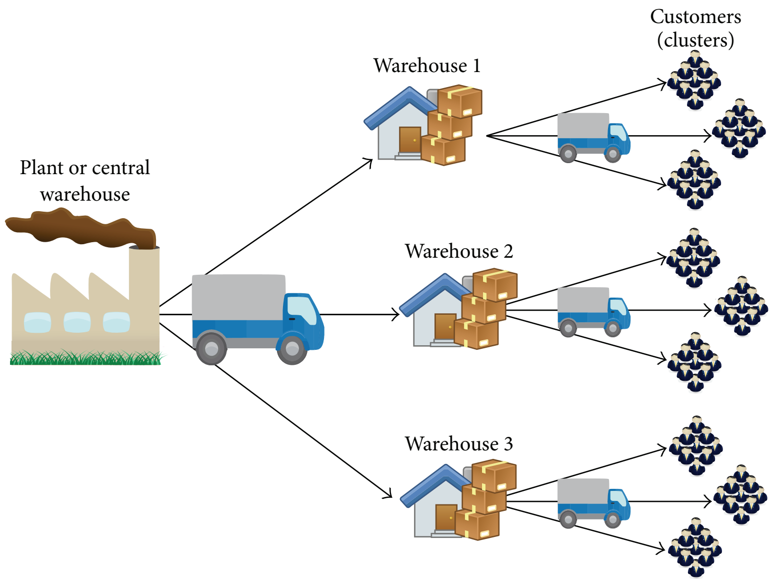

Figure 1: Distribution network structure considered in this study. It consists of one central plant, a set of possible warehouses, and a set of customers or retailers.

$$Y_{ij} \leq X_i \quad \forall i = 1, 2, \ldots, N, \ \forall j = 1, 2, \ldots, M \tag{4}$$

$$X_i, Y_{ij} \in \{0, 1\} \quad \forall i = 1, \ldots, N; \ \forall j = 1, \ldots, M. \tag{5}$$

Equation (1) is the total system cost. The first term is the fixed set-up and operating cost when opening warehouses. The second term is the daily transport cost between warehouse and customers which depends on the customer demand d and distance C_{ij} between warehouse i and customer j. Inequality (2) ensures that total demand of warehouse i will never be greater than its capacity I^{cap}. Equation (4) ensures that customers are served by only one warehouse. Finally, (5) states (0-1) condition for the binary variables Y_{ij} and X_i.

While both heuristics and mathematical programming methods have been used to solve the CFLP, in this work we present a *matheuristic* method [1] that combines two well-known algorithms: mixed-integer programming and local search. During the last decade, the idea of combining the power of mathematical programming with flexibility of heuristics has gained attention within researchers community. We can find matheuristics attempting to solve problems arising in the field of logistics [2–4], health care systems [5–7], and pure mathematics [8, 9], among others. Matheuristics have been demonstrated to be a promising research field in order to solve complex optimisation problems. Some interesting surveys on matheuristics are found in [10, 11].

The remainder of the paper is organised as follows. First, a literature review on the techniques for solving the CFLP is presented in Section 2. Then, in Section 3, we describe the matheuristic approach used in this work. Section 4 starts explaining the procedure used to generate the set of instances presented in this paper. In Section 4.1, the obtained results are presented. Finally, in Section 5, some conclusions based on the numerical results are outlined.

2. Literature Review

The CFLP is a well-known problem in operations research. Heuristic methods, such as evolutionary algorithms [12–14] and local search [15, 16], as well as mathematical programming [17–19] have been used to address this problem. Several surveys have been dedicated to cover this problem and its variations [20, 21]. In spite of that, only little attention has been paid to matheuristics attempting to solve the CFLP problem. In [22], the authors develop a Lagrangian-based heuristic (LH) that provides lower bounds to the problem, and a Tabu Search (TS) algorithm is subsequently used to find the upper bound of the problem. In this case, the TS is initialised using the primary information provided by LH. Additionally, in [23], the authors combine Lagrangian Relaxation with Ant Colony Optimisation to solve the discrete CFLP problem. Other authors have presented matheuristic approaches applied to CFLP variations such as the $(r \mid p)$-centroid problem [24] and p-median problem [25]. None of the approaches mentioned before is applied to large-scale CFLP problems. In [26], authors propose hybrid algorithm that combines artificial bee algorithms and mixed-integer programming (MIP) to solve large-scale CFLP problems ranging from 300 to 1000 warehouses and 1000 customers. They found that their hybrid approach outperformed each technique separately. This is particularly true for largest instance in their study.

3. Proposed Approach

Heuristic methods are a common approach to solve hard combinatorial optimisation problems such as the CFLP. Despite the fact that heuristics do not guarantee optimality, the solutions provided by them can be considered good suboptimal ones. In contrast exact methods guarantee optimality; however, they usually fail when dealing with medium- and large-sized problems. In this paper, a local search algorithm, which is a variation of the well-known Tabu Search (TS) algorithm, has been developed.

3.1. Local Search Framework. As we mentioned above, the proposed matheuristic algorithm is based on a local search strategy which in turn is a variation of the well-known TS algorithm. As in TS we make use of an adaptive memory structure, namely, tabu list. Moreover, our algorithm, as in any local search algorithm, needs to "exchange information" with its neighbours. To do that, first, the neighbourhood must be defined. Our algorithm has only one neighbourhood move that defines a set of possible candidate solutions. This move is used across all executions of the algorithm. Moving from one solution to one of its neighbours implies that we need to close k of the open facilities and, at the same time, open k of the previously closed one. That means the size of the neighbourhood is $\mathcal{O}\left(\binom{k}{W^+}\times\binom{k}{W^-}\right)$, where W^+ is the number of open facilities in the current solution and W^- is the number of closed facilities. The number of facilities that are open at each iteration remains constant. Different rules could be used in order to decide what facilities should be opened/closed at each iteration. In this paper such a decision is made randomly, taking into account the fact that the total installed capacity, that is, the sum of the individual capacities over the set of opened facilities, must be larger than the total demand of the system +10%. Thus, we ensure that feasible allocations can be found for the new set of open facilities and that the capacity constraint shall not be violated.

Clearly, other neighbourhood definitions might be considered. For instance, using neighbourhood definitions that lead to smaller neighbourhoods could be considered, as it would allow us to explore the entire neighbourhood enhancing the exploitation ability of our algorithm. However, smaller neighbourhood definition might impair the exploration ability of the algorithm. Thus, although the proposed algorithm supports different neighbourhood definitions, changing it might provoke a big impact in the algorithm behaviour.

Moreover, since we want to solve large instances of the CFLP problem, visiting the entire neighbourhood for the current solution is impractical in terms of computational time. Thus, we need to select a subset of the neighbourhood of the current solution. After an extensive trial and error process, we end up with the size of the neighbourhood set equal to 10. We need to stress that this number largely depends on the available computational resources. Thus, it might not be the best one for other researchers using different infrastructure.

Our algorithm also shared with TS a diversification mechanism that allows it to get out of low-quality neighbourhoods and "jump" to explore new neighbourhoods. The diversification mechanism implemented here is a restart method, which reinitialises a current solution without losing the best solution found by the algorithm.

One distinctive difference between our LS algorithm and TS is that in our approach we do not start with a (completely) random solution but instead we try to find a "warm start" solution by (partially) solving a subproblem that includes a large portion of the available facilities (columns). We use MIP solver to solve the subproblem described above and set a very short time-out (15 secs). After the solver finishes because of the time-out we keep those facilities that are open and discard the closed ones. It is important to note that the number of columns considered in the first iteration is large enough to provide a warm start solution and, at the same time, small enough so the solver can converge within the time-out. Discarded facilities are discarded only regarding the initial solution. Thus, after we found the warm start, all the possible facilities are considered for next iterations.

```
begin
    $k = 0$;
    $s_k$ = initialSolution($N$);
    $s_{best} = s_k$;
    while within timeLimit do
        $NS_k$ = generateNeighbourhood($s_k$);
        $ns_k^{best}$ = findBest($NS_k$);
        while $ns_k^{best}$ isTabu do
            if cost of $ns_k^{best}$ < cost of $s_{best}$ then
                aspirationCriterion;
                break;
            replace $ns_k^{best}$;
        if $f(ns_k^{best}) < f(s_{best})$ then
            $s_{best} = ns_k^{best}$;
            noBestSol = 0;
        noBestSol++;
        $s_k = ns_k^{best}$;
        update(tabuList);
        check (diversificationCriterion, noBestSol);
        $k = k + 1$
```

Algorithm 1: LS-MIP matheuristic algorithm.

3.2. Proposed Matheuristic Approach. In this work we propose a matheuristic approach which combines a local search (LS) algorithm and a mixed-integer programming solver. The LS algorithm starts by obtaining a warm start solution as described above. Once the algorithm has set open warehouses, the MIP solver is called to solve the subproblem that considers only the open warehouses as candidates. That means that N becomes W^+ in (1), (2), (3), (4), and (5), where $W^+ < N$. This allows us to fairly compare different sets of possible warehouses as we obtain the optimal solution for each subproblem. We need to state at this point that the number of open warehouses W^+ at each iteration is very important and, consequently, we need to choose it carefully. On one hand, setting W^+ too large would lead to a subproblem that is not possible to solve to optimality within an appropriate time. On the other hand, setting W^+ too small would provoke the subproblem to become infeasible as there is no enough installed capacity to serve the total demand from customers. Algorithm 1 shows the main steps of our approach.

Once the initial solution is calculated, the neighbourhood NS_k is generated. The corresponding MIP subproblem is solved for each neighbour. We then select the neighbour with the lowest cost and check whether it is in the tabu list or not. In case it is in the tabu list, we check whether it is better than the best solution found so far. If so, the new solution (aspiration criterion in TS) is set. Otherwise, the second best neighbour is checked.

Once a neighbour is selected from the neighbourhood list we check whether the new best neighbour is better than the best solution so far. If so, we reset the diversification counter. Otherwise, the diversification counter increases in one. If the diversification counter reaches its threshold, the restart method is called and a new solution is generated as in the first iteration. We perform some few experiments to find a good value for the threshold. As expected, we found that threshold value depends on the size of the problem. For small cases, threshold is set between 8 and 12 iterations without improvement. For larger instances, threshold is set to 30. The algorithm proceeds in the same way until the time limit is reached. In our case, time limit was set at 2000 secs for all our experiments.

Table 1: Obtained results for instances with 500 customers and 500 to 1000 possible warehouse locations.

		TotalCost	*Time* (sec)	*k*	*GAP* (%)
500 × 500	MIP	97038	2002	0	**0.00**%
	RAND	103546	1101	79	6.71%
	MATH	97273	1276	21	0.24%
600 × 500	MIP	116388	2002	0	0.06%
	RAND	127667	1182	108	9.75%
	MATH	116321	1499	25	**0.00**%
700 × 500	MIP	138396	2005	0	0.15%
	RAND	153628	1505	284	11.17%
	MATH	138190	1670	26	**0.00**%
800 × 500	MIP	158222	2003	0	0.45%
	RAND	178445	675	167	13.29%
	MATH	157518	1110	19	**0.00**%
900 × 500	MIP	179087	2003	0	0.48%
	RAND	199431	238	34	11.90%
	MATH	178230	1008	18	**0.00**%
1000 × 500	MIP	202706	2003	0	0.45%
	RAND	214914	2228	102	6.50%
	MATH	201794	976	15	**0.00**%

4. Computational Experiments

In this section, we present the benchmark applied for performance comparison and a summary of the computational results obtained for our matheuristic approach. Computational experiments were performed on an Intel Core Duo processor CPU T2700, 2.33 GHz, with 6 GB of RAM. Linux 14.02 was the operating system. The matheuristic algorithm was coded in JAVA 8 language using NetBeans IDE. The MIP subproblem was modelled using AMPL and solved with GUROBI 6.0 solver.

To validate the algorithm and measure its convergence, we chose a set of medium-sized instances that have known optimal solutions, namely, capa, capb, and capc. These instances were obtained from Beasley's OR-Library (http://people.brunel.ac.uk/~mastjjb/jeb/info.html). Optimal solutions for these instances can be found in [17]. Our algorithm found the optimal solution for all these instances within acceptable times. We then create a set of large instances with 500 to 1000 warehouses (step size of 100) and 500 clients using the strategy provided in [27]. This strategy was also used in [26]. 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 j and a warehouse i corresponds to the transportation cost C_{ij}. The demand d_j is calculated using a uniform distribution $\mathscr{U}[5,35]$. I_i^{cap} is calculated using $\mathscr{U}[100,1600]$. We amplify the capacity of the warehouses to obtain harder instances. Finally, the fixed cost of warehouse i is $F_i = \mathscr{U}[0,90] + \mathscr{U}[100,110] \times \sqrt{I_i^{\text{cap}}}/10$. We generate 6 different classes of problems: $\{500,600,700,800,900,1000\} \times \{500\}$ (warehouses × customers). To avoid any instance-dependent effects, we generate 10 different instances for each class. In order to compare our algorithm with previously proposed algorithms in the literature, we apply our algorithm to three sets of instances proposed by Avella and Boccia [28] and also used by Guastaroba and Speranza [29]. We call these three sets of instances *AB1, AB2,* and *AB3*, where AB1 consists of 20 test instances with 300 potential warehouses and 300 clients. AB2 also consists of 20 instances with 500 potential warehouses and 500 clients. Finally, AB3 consists of 20 instances with 700 potential warehouses and 700 clients. We execute our algorithm 30 times for each instance to assess and avoid outlier performance. Results we present later in this section correspond to the average values obtained for each class of problems. The MIP algorithm was executed only once per instance due to its deterministic behaviour.

4.1. Results.

4.1. Results. In this subsection we present a summary of the obtained results. Table 1 shows all the obtained results.

First column denotes the instance name. Second column corresponds to the algorithm by which the results were obtained. Here *MIP* means the mathematical programming approach using the MIP solver from GUROBI, *RAND*, corresponds to our baseline algorithm which allocates customers randomly. *MATH* corresponds to our matheuristic approach. Column *TotalCost* corresponds to the average total cost obtained by each approach. Column *Time* corresponds to average time the algorithm takes to find the best solution, while k column shows the iteration where the best solution was found by each algorithm (in average). Column *GAP* shows the difference between the result obtained by the corresponding algorithm and the best solution known for each instance.

As we can see in Table 1, as the number of decision variables increases our approach obtains better results with respect to the ones obtained by the LS and MIP separately. The MIP solver obtains better results than our matheuristic

TABLE 2: Results for the AB instances.

	AB1				AB2				AB3		
#	*GAP*	*BestGAP*	*t* (sec)	#	*GAP*	*BestGAP*	*t* (sec)	#	*GAP*	*BestGAP*	*t* (sec)
1	0.00%	0.00%	84.0	1	0.00%	0.00%	174.8	1	0.00%	0.00%	201.3
2	0.00%	0.00%	95.6	2	0.00%	0.00%	112.2	2	0.00%	0.00%	311.5
3	0.00%	0.00%	141.0	3	0.00%	0.00%	166.2	3	0.00%	0.00%	213.1
4	0.00%	0.00%	182.3	4	0.00%	0.00%	193.6	4	0.00%	0.00%	226.4
5	0.00%	0.00%	83.0	5	0.00%	0.00%	554.2	5	0.00%	0.00%	543.7
6	0.00%	0.00%	159.3	6	0.00%	0.00%	96.4	6	0.00%	0.00%	827.3
7	0.01%	0.00%	211.8	7	0.00%	0.00%	146.7	7	0.12%	0.09%	1228.5
8	0.00%	0.00%	158.0	8	0.07%	0.00%	713.6	8	0.00%	0.00%	154.7
9	0.00%	0.00%	31.7	9	0.00%	0.00%	166.9	9	0.00%	0.00%	247.8
10	0.10%	0.10%	16.0	10	0.00%	0.00%	69.8	10	0.90%	0.10%	695.4
11	0.13%	0.11%	74.7	11	0.47%	0.31%	796.7	11	0.00%	0.00%	608.5
12	0.00%	0.00%	77.8	12	0.20%	0.17%	951.3	12	0.17%	0.17%	188.0
13	0.17%	0.15%	133.5	13	0.00%	0.00%	269.7	13	0.00%	0.00%	163.1
14	0.00%	0.00%	118.3	14	0.00%	0.00%	363.6	14	0.00%	0.00%	63.2
15	1.02%	0.77%	64.1	15	0.00%	0.00%	463.2	15	0.09%	0.09%	426.4
16	0.00%	0.00%	109.7	16	0.09%	0.00%	336.7	16	0.11%	0.08%	795.8
17	0.00%	0.00%	27.4	17	0.13%	0.09%	318.5	17	0.11%	0.11%	59.6
18	0.34%	0.28%	449.0	18	0.08%	0.00%	962.7	18	0.00%	0.00%	75.9
19	0.13%	0.11%	90.6	19	0.08%	0.00%	933.1	19	0.00%	0.00%	66.7
20	0.00%	0.00%	108.5	20	0.18%	0.18%	361.8	20	0.21%	0.16%	118.9

approach only for the smallest class (500 × 500). For all the other problems, our matheuristic approach improves results obtained by the MIP algorithm and performs faster. We need to point out that, because of the time limit we impose, the MIP solver is not always able to find the optimal solution within the allowed time. Figure 2 shows convergence of both MIP and matheuristic algorithms. Crosses correspond to the matheuristic algorithm. Circles correspond to the MIP algorithm and triangles correspond to the lower bound obtained by the MIP algorithm.

As we can see in Figure 2, our algorithm converges much faster than MIP does. Moreover, Figure 2 suggests that diversification methods other than the random one used in this work could be very useful to avoid that the matheuristic algorithm which gets trapped into poor locally optimal solutions.

Regarding AB instances, Table 2 shows a summary of the obtained results for such instances. Column # is the corresponding instance number. Columns *GAP* and *BestGAP* show the average GAP after 10 runs and the best GAP obtained, respectively. Time (t) is shown in seconds. As we can see, our algorithm is able to obtain the optimal solution (GAP = 0) for most of the instances. Moreover, the average GAP equal or close to the best obtained GAP is a good indicator of the reliability of the algorithm. Although results are good in terms of the obtained GAP values, other algorithms in the literature, such as the Kernel Search in Guastaroba and Speranza [29], exhibit a better convergence rate. Thus, it is worth to further study the behaviour of our algorithm in order to find alternative ways to speed it up.

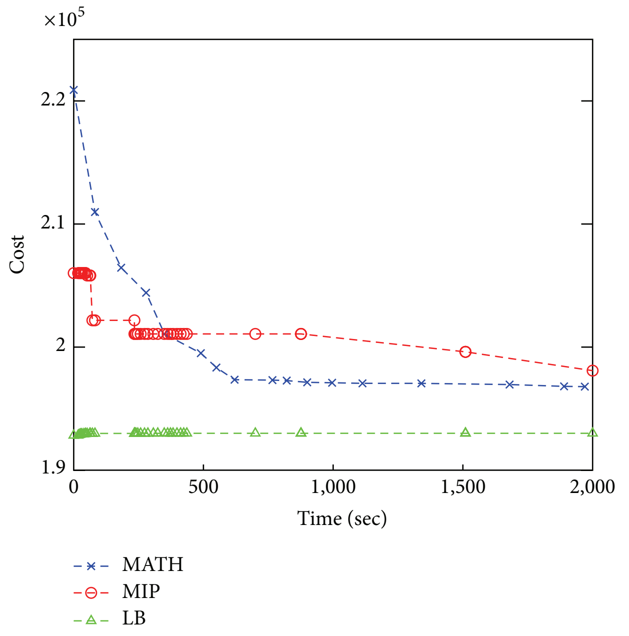

FIGURE 2: Convergence of both MIP and matheuristic algorithm.

5. Conclusions and Future Work

In this paper, we have presented a matheuristic approach that is able to find good solutions within acceptable time. Our algorithm combines the flexibility of a local search framework based on the well-known Tabu Search metaheuristic and the calculation efficiency of the exact algorithm embedded within the MIP solver. Our algorithm solves many subproblems by opening and closing, in turns, available warehouse locations. Once a new set of candidates locations is set, the MIP solver solves to optimality the associated allocation subproblem. In average, our algorithm is able to find solutions that are below the upper bound found by the MIP algorithm and, therefore, is able to minimize the gap between the best solution found and the corresponding lower bound found by the exact method. Furthermore, our algorithm converges faster than the associated MIP solver.

As a future work, new heuristic methods can be used to seek good combinations of possible warehouse locations. Moreover, other exact methods such as Lagrangian Relaxation can be tested in order to speed up subproblem solving process. Also new diversification strategies as well as guided search should be studied.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

Fernando Paredes is supported by FONDECYT-Chile (Grant no. 1130455).

References

[1] V. Maniezzo, T. Stützle, and S. Voß, Eds., *Matheuristics—Hybridizing Metaheuristics and Mathematical Programming*, vol. 10 of *Annals of Information Systems*, Springer, New York, NY, USA, 2010.

[2] M. Caserta and S. Voss, "A math-heuristic Dantzig-Wolfe algorithm for capacitated lot sizing," *Annals of Mathematics and Artificial Intelligence*, vol. 69, no. 2, pp. 207–224, 2013.

[3] L. C. Coelho, J.-F. Cordeau, and G. Laporte, "Heuristics for dynamic and stochastic inventory-routing," *Computers & Operations Research*, vol. 52, part A, pp. 55–67, 2014.

[4] B. Raa, W. Dullaert, and E.-H. Aghezzaf, "A matheuristic for aggregate production-distribution planning with mould sharing," *International Journal of Production Economics*, vol. 145, no. 1, pp. 29–37, 2013.

[5] H. Allaoua, S. Borne, L. L'etocart, and R. Calvo, "A matheuristic approach for solving a home health care problem," *Electronic Notes in Discrete Mathematics*, vol. 41, pp. 471–478, 2013.

[6] Y. Li, D. Yao, W. Chen, J. Zheng, and J. Yao, "Ant colony system for the beam angle optimization problem in radiotherapy planning: a preliminary study," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1532–1538, Edinburgh, Scotland, September 2005.

[7] Y. Li, D. Yao, J. Zheng, and J. Yao, "A modified genetic algorithm for the beam angle optimization problem in intensity-modulated radiotherapy planning," in *Artificial Evolution*, E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, and M. Schoenauer, Eds., vol. 3871 of *Lecture Notes in Computer Science*, pp. 97–106, Springer, Berlin, Germany, 2006.

[8] S. Hanafi, J. Lazić, N. Mladenović, C. Wilbaut, and I. Crévits, "New hybrid matheuristics for solving the multidimensional knapsack problem," in *Hybrid Metaheuristics*, M. Blesa, C. Blum, G. Raidl, A. Roli, and M. Sampels, Eds., vol. 6373 of *Lecture Notes in Computer Science*, pp. 118–132, Springer, Berlin, Germany, 2010.

[9] E.-G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *4OR*, vol. 11, no. 2, pp. 101–150, 2013.

[10] C. Archetti and M. G. Speranza, "A survey on matheuristics for routing problems," *EURO Journal on Computational Optimization*, vol. 2, no. 4, pp. 223–246, 2014.

[11] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. Bolufé Röhler, "Matheuristics: optimization, simulation and control," in *Hybrid Metaheuristics*, M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, and A. Schaerf, Eds., vol. 5818 of *Lecture Notes in Computer Science*, pp. 171–177, Springer, Berlin, Germany, 2009.

[12] 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 and Industrial Engineering*, vol. 75, no. 1, pp. 200–208, 2014.

[13] 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.

[14] M. J. Cortinhal and M. E. Captivo, "Genetic algorithms for the single source capacitated location problem," in *Metaheuristics: Computer Decision-Making*, vol. 86 of *Applied Optimization*, pp. 187–216, Springer, New York, NY, USA, 2004.

[15] I. A. Contreras and J. A. Díaz, "Scatter search for the single source capacitated facility location problem," *Annals of Operations Research*, vol. 157, no. 1, pp. 73–89, 2008.

[16] C. Valenzuela, B. Crawford, R. Soto, E. Monfroy, and F. Paredes, "A 2-level metaheuristic for the set covering problem," *International Journal of Computers, Communications and Control*, vol. 7, no. 2, pp. 377–387, 2012.

[17] J. E. Beasley, "An algorithm for solving large capacitated warehouse location problems," *European Journal of Operational Research*, vol. 33, no. 3, pp. 314–325, 1988.

[18] J. A. Diaz and E. Fernández, "A branch-and-price algorithm for the single source capacitated plant location problem," *Journal of the Operational Research Society*, vol. 53, no. 7, pp. 728–740, 2002.

[19] Z. Yang, F. Chu, and H. Chen, "A cut-and-solve based algorithm for the single-source capacitated facility location problem," *European Journal of Operational Research*, vol. 221, no. 3, pp. 521–532, 2012.

[20] J. R. Current, M. Daskin, and D. Schilling, *Discrete Network Location Models*, chapter 3, Springer, 2002.

[21] Z. Drezner, *Facility Location: A Survey of Applications and Methods*, Springer, New York, NY, USA, 2011.

[22] M. Cortinhal and M. 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.

[23] C.-H. Chen and C.-J. Ting, "Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 6, pp. 1099–1122, 2008.

[24] E. Alekseeva and Y. Kochetov, "Matheuristics and exact methods for the discrete (rp)-centroid problem," in *Metaheuristics for Bi-level Optimization*, E.-G. Talbi, Ed., vol. 482 of *Studies in Computational Intelligence*, pp. 189–219, Springer, Berlin, Germany, 2013.

[25] M. Yaghini, M. Karimi, and M. Rahbar, "A hybrid metaheuristic approach for the capacitated p-median problem," *Applied Soft Computing*, vol. 13, no. 9, pp. 3922–3930, 2013.

[26] G. Cabrera, E. Cabrera, R. Soto, L. J. M. Rubio, B. Crawford, and F. Paredes, "A hybrid approach using an artificial bee algorithm with mixed integer programming applied to a large-scale capacitated facility location problem," *Mathematical Problems in Engineering*, vol. 2012, Article ID 954249, 14 pages, 2012.

[27] 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.

[28] P. Avella and M. Boccia, "A cutting plane algorithm for the capacitated facility location problem," *Computational Optimization and Applications*, vol. 43, no. 1, pp. 39–65, 2009.

[29] G. Guastaroba and M. G. Speranza, "Kernel search for the capacitated facility location problem," *Journal of Heuristics*, vol. 18, no. 6, pp. 877–917, 2012.