Hindawi
Advances in Operations Research
Volume 2020, Article ID 5239176, 7 pages
https://doi.org/10.1155/2020/5239176

*Research Article*

# Solving Capacitated Facility Location Problem Using Lagrangian Decomposition and Volume Algorithm

**Eiman J. Alenezy**

*Department of Mathematics, College of Basic Education Public Authority for Applied Education and Training Kuwait, Kuwait City, Kuwait*

Correspondence should be addressed to Eiman J. Alenezy; ej.alenezy@paaet.edu.kw

Received 21 October 2019; Revised 10 December 2019; Accepted 30 December 2019; Published 4 February 2020

Academic Editor: Panagiotis P. Repoussis



In this research, we will focus on one variant of the problem: the capacitated facility location problem (CFLP). In many formulations of the CFLP, it is assumed that each demand point can be supplied by only one open facility, which is the simplest case of the problem. We consider the case where each demand point can be supplied by more than one open facility. We first investigate a Lagrangian relaxation approach. Then, we illustrate in the problem decomposition how to introduce tighter constraints, which solve the CFLP faster while achieving a better quality solution as well. At the same time, we apply the volume algorithm to improve both the lower and the upper bound on the optimum solution of the original problem for the large problem size.

## 1. Introduction

There are a limited number of test case problems for the CFLP that we know of in the literature, which can be used to evaluate the results of our main goal of this research, the Lagrangian decomposition algorithm. Most of the algorithms developed in the literature have been tested on simulated data.

Lagrangian relaxation or Lagrangian decomposition was introduced in the early 70's through the work of Held and Karp [1, 2] and Held et al. [3]. They realized that the relationship between the systematic traveling-salesman problem and the minimum spanning tree problem gives a sharp lower bound on the optimum solution. Thereafter, they used a procedure called subgradient optimization [3] and showed that it is effective for approximating the maximum of certain piecewise linear concave functions. The concept of Lagrangian relaxation is that we relax some constraints and then penalize their violation, and the motivation for the relaxation of these constraints is that many combinatorial optimization problems consist of an easy problem with some additional complicating constraints. Applying the Lagrangian relaxation to these problems will relax these complicating constraints and absorb them into the objective function. The resulting relaxed problem becomes much easier to solve.

Barahona and Anbil [4] presented an extension to the subgradient algorithm that will produce an approximation cost per iteration; they called it the volume algorithm. In general, it produces a primal vector as well as a dual vector that can be used as a starting point for a more exact method. They were able to present a successful experiment with linear programming problems.

A large number of approximation algorithms with different techniques have been proposed recently for various applications of the CFLP; for example, refer [5–11].

Alenezy and Khalaf in [12] discussed the Lagrangian decomposition and volume algorithm procedures in detail. In this article, we use variants of it in solving our CFLP for large size problems. The results obtained in this work were compared with the results of best greedy and greedy weak solutions.

In order to evaluate the performance of our Lagrangian decomposition algorithm, we develop a number of greedy heuristics (filters) that provide good solutions for known problems in the literature [9]. This is to illustrate the

effectiveness of these heuristics for comparative purposes when evaluating our Lagrangian decomposition approach on large problem instances.

In the extreme case of very large problems, it is not possible to have any strong constraints. However, we take a dynamic approach to the model representation, whereby an effective tightening constraint is dynamically added to the model representation.

In particular, in this paper, we illustrate all the results and the analysis of our experiments for both the greedy heuristics and the Lagrangian decomposition algorithm. We also include tests on the performance of our algorithm compared to a number of benchmark models.

## 2. Capacitated Facility Location Problem

The facility location problem (FLP) seeks to locate a number of facilities to serve a number of customers; thus, there is a set of potential facility locations $F$; opening a facility at location $i \in F$ has an associated nonnegative fixed cost $f_i$ and has either a limited or unlimited capacity $S_i$ of available supply. There is a set of customers (clients) or demand points $D$ that require service; customer $j \in D$ has a demand $d_j$ that must be satisfied by the open facilities. If a facility at location $i \in F$ is used to satisfy part of the demand of client $j \in D$, then there is a service or transportation cost incurred, which is often proportional to the distance from $i$ to $j$, denoted by $c_{ij}$.

Let $F$ = a set of potential facility locations, $D$ = a set of customers or demand points, $m$ = the number of potential locations of the facilities; $m = |F|$, $n$ = the number of customer; $n = |D|$, $j_d$ = the demand of customer $j \in D$(where $d_j \geq 0$), $c_{ij}$ = the unit cost of supplying the demand of customer $j \in D$ from facility $i \in F$ (where $c_{ij} \geq 0$), $S_i$ = the capacity of facility $i$ (i.e., the upper limit on the total demand that can be supplied from facility $i$, where $S_i \geq 0$), $p$ = the desired number of open facilities, $f_i$ = the fixed cost associated with opening facility $i$ (where $f_i \geq 0$), $x_{ij}$ = the fraction of the demand of customer $j$ supplied from facility $i$ (where $0 \leq x_{ij} \leq 1$), and $y_i = \begin{cases} 1 & \text{if facility } i \text{ is open} \\ 0 & \text{otherwise} \end{cases}$.

Then, the formulation of the CFLP as a mixed-integer programming problem, which is referred to as IP found in [12], is as follows.

The objective equation:

$$\begin{aligned} &\text{minimize} \quad \sum_{j\in D}\sum_{i\in F} d_j c_{ij} x_{ij} + \sum_{i\in F} f_i y_i, \\ &\text{subject to} \quad \sum_{i\in F} x_{ij} = 1, \quad \forall j \in D. \end{aligned} \tag{1}$$

This equation ensures that the demand of each customer is satisfied:

$$\sum_{j\in D} d_j x_{ij} \leq S_i y_i, \quad \forall i \in F, \tag{2}$$

$$x_{ij} \leq y_i, \quad \forall i \in F,\ j \in D. \tag{3}$$

To ensure that the closed facility does not supply any customer and that the demand supplied from the facility does not exceed the capacity of the facility,

$$y_i \in \{0, 1\}, \quad \forall i \in F. \tag{4}$$

This is the integrality constraint:

$$p \leq \sum_{i\in F} y_i \leq p + 2. \tag{5}$$

This constraint tightens the lower bound representation of the CFLP. The value of $p$ is determined as follows; we sort all of the facilities in descending order of capacity. Then, $p$ is such that, in the sorted order of facilities,

$$\sum_{i=1}^{p-1} S_i < \sum_{j=1}^{n} d_i \leq \sum_{i=1}^{p} S_i. \tag{6}$$

Although the right-hand side of the constraint (5) need not be bounded, as a consequence of our experiments using $p + 2$ speeds up considerably the computational time with no detriment to the solution quality,

$$x_{ij} \geq 0, \quad \forall i \in F,\ j \in D. \tag{7}$$

The last inequality provides bounds on the allocation variables $x_{ij}$.

In order to develop a Lagrangian heuristic for the CFLP, first we need to consider a linear programming relaxation for the IP problem, which is the same formulation (IP), except we replace inequalities (4) by

$$y_i \leq 1, \quad \forall i \in F. \tag{8}$$

We will denote the LP relaxation by $P$.

## 3. Lagrangian Decomposition and Volume Algorithm (LD and VA)

In this part, we first consider a Lagrangian relaxation of the above problem ($P$). Then, we describe how to use the volume algorithm [7], which is an extension to the subgradient optimization.

In order to investigate the solution of large-sized problems, we followed the methods of Alenezy and Khalaf [12], That is, by decomposing CFLP into independent problems, which are easier to solve. Motivated by this method, let $u_i$ be the dual multiplier for the equation $j$ in (1) and let $\bar{c}_{ij} = d_j c_{ij} - u_j$. Then, a lower bound $L(u)$ is given by solving the following problem:

$$L(u) = \text{minimize} \sum_{j\in D}\sum_{i\in F} \bar{c}_{ij} x_{ij} + \sum_{i\in F} f_i y_i, \tag{9}$$

$$\text{subject to} \quad \sum_{j\in D} d_j x_{ij} \leq S_i y_i, \quad \forall i \in F, \tag{10}$$

$$x_{ij} \leq y_i, \quad \forall i \in F,\ j \in D, \tag{11}$$

$$y_i \leq 1, \quad \forall i \in F, \tag{12}$$

$$p \le \sum_{i \in F} y_i \le p + 2, \tag{13}$$

$$x_{ij} \ge 0, \quad \forall i \in F,\ j \in D. \tag{14}$$

*3.1. The Lagrangian Decomposition (LD).* It was widely reported that solving the $L(u)$ above provides a good lower bound on the integer optimum solution. This is improved by using the volume algorithm. Motivated by this and by the fact that it is difficult to solve the $L(u)$ for large size problems, we decompose the $L(u)$ problem into $m$ independent subproblems for each $i \in F$ to compute a lower bound, we also at this point relax constraint (13), and we will return to these later in this section. The generic forms of each subproblem are

$$\begin{aligned} &\text{minimize} && f y + \sum \overline{c}_j x_j, \\ &\text{subject to} && \sum_j d_j x_j \le S y, \\ & && x_j \le y,\ j \in D, \\ & && 0 \le y \le 1, \\ & && x \ge 0. \end{aligned} \tag{15}$$

Solving this to get a lower bound (LB) is easy, and it is easy too for the upper bound (UB). First, we set to 0 any variable $x_j$ with $\overline{c}_j > 0$. Then, we assume that the rest of the variables are ordered such that

$$\frac{\overline{c}_1}{d_1} \le \frac{\overline{c}_2}{d_2} \le \cdots \frac{\overline{c}_{n'}}{d_{n'}}, \quad \text{where } n' \subseteq n \text{ and } n = |D|. \tag{16}$$

Now, let $k$ be the largest index such that $\sum_{j=1}^{k} d_j \le S$, and let

$$\begin{aligned} b(k) &= \sum_{j=1}^{k} d_j, \\ r &= \frac{S - b(k)}{d_{k+1}}. \end{aligned} \tag{17}$$

If $f + \sum_{j=1}^{k} \overline{c}_j + \overline{c}_{k+1} r \ge 0$, then we set $y = 0$ and $x_{ij} = 0$ for all $j$. Otherwise, we set $y = x = 1$ for $1 \le j \le k$ and $x_{k+1} = r$.

Having solved these $m$ independent subproblems, we next consider the minimum number of facilities that are needed to supply all the demand. In this way, we enhance the LB here by comparing the number of opened facilities denoted by $h$, after having solved the $m$ subproblems, to the minimum needed $p$. If $h < p$, then we sort the unopened facilities by their fixed costs and open the cheapest facilities until we have $p$ opened. The LB is suitably increased to account for these extra fixed costs.

*3.2. The Volume Algorithm (VA).* To improve the lower bound we obtained from solving the decomposition of the problem above, we use the VA developed in [7] and the same algorithm found in [12], which stopped by a number of passes = 200, and next, we extend it to 2000. This algorithm can be formulated by the following steps:

*Step 1.* Start with a vector $\overline{u}$ and solve (8)–(13) to obtain $(\overline{x}, \overline{y})$ and $L(\overline{u})$ and set $t = 1$.

*Step 2.* Compute $v^t$, where $v_j^t = 1 - \sum_i \overline{x}_{ij}$ and $u^t = \overline{u} + s v^t$, for a step size $s$ given by (20) below. Solve (8)–(13) with $u^t$. Let $(x^t, y^t)$ be the solution obtained. Then, $(\overline{x}, \overline{y})$ is updated as

$$(\overline{x}, \overline{y}) = \alpha\left(x^t, y^t\right) + (1 - \alpha)(\overline{x}, \overline{y}), \tag{18}$$

where $\alpha$ is a number between 0 and 1. In order to set the value of $\alpha$, we solve the following one-dimensional problem:

$$\begin{aligned} &\text{minimize} && \|\alpha w + (1 - \alpha) v^t\|, \\ &\text{subject to} && \frac{b}{10} \le \alpha \le b. \end{aligned} \tag{19}$$

The value $b$ was originally set to 0.1, and then every 100 iterations we check if $L(u^t)$ had not increased by at least 1%, in which case, we divide $b$ by 2; otherwise, we keep it as it is. When $b$ becomes less than $10^{-5}$, we keep it constant at this value.

*Step 3.* If $L(u^t) > L(\overline{u})$, then update $\overline{u}$ as $\overline{u} = u^t$. Notice that in, step 3, we update $\overline{u}$ only if $L(u^t) > L(\overline{u})$.

*Step 4.* Stopping criteria.

(1) $v_j^t < 0.02$
(2) $(\text{UB} - L(u^t))/\text{UB} \le 0.02$
(3) The number of passes equals 2000

The algorithm terminates when one of the previous criteria is met. If stopping rules are not satisfied, then set $t = t + 1$, and go to step 2.

The formula of the step size $s$ is as the one used in the subgradient method [13]:

$$s = \lambda \frac{\text{UB} - L(\overline{u})}{\|v\|^2}, \tag{20}$$

where $\lambda$ is a number between 0 and 2. In order to set its value, we define three types of iterations:

(i) Iteration $E$, which is the iteration with no improvement on the lower bound. A sequence of $E$ iterations requires the need for a smaller step size. Therefore, after a sequence of 20 $E$ iterations, we multiply $\lambda$ by 0.66.

(ii) If $L(u^t) > L(\overline{u})$, we compute $w_j = \sum_i x_{ij}^t$, for all $j$ and $d = v^t \cdot w$. If $d < 0$, this means that a larger step size would have given a smaller value for $L(u^t)$, and we call this iteration $Y$.

(iii) If $d \ge 0$, we call this iteration $T$; a $T$ iteration suggests the need for a larger step size, so we multiply $\lambda$ by 1.1.

*3.3. Computing LB.* In [12], we found that there are two approaches to solve the $L(u)$ formulation of the problem above to obtain a LB. To keep this paper self-contained, we list the following approaches:

The first approach is to decompose $L(u)$ into $m$ independent subproblems. Solve them as explained above. Then, update this LB using the VA as mentioned previously, but in two ways for step 1. One way is to set the value of the vector $\overline{u}$ to 0 and continue using the VA to improve the LB. The other way is to set the vector $\overline{u}$ to values that is denoted by "warm start duals." The warm start duals are the values of the duals of the relaxed constraints (1), obtained from solving the greedy weak representation of the CFLP as a solution.

The second approach is first to remove constraints (11) from the $L(u)$ formulation to reduce the size of the problem to make it possible to solve. Then, solve it without the decomposition technique above, to obtain a LB. To improve this LB again, we apply the VA with the same two ways we discussed above.

Notice that the constraints (13) are included in the $L(u)$ formulation, but only the part $p \leq \sum_{i \in F} y_i$ is applicable in obtaining the LB as we explained before. The other part is redundant here since it only has an impact on obtaining the UB in the next approaches. The lower bound in this approach does not always have integer values $y$, while this is not the case in the decomposition approach; hence, the LB from this approach is often worse than the LB from the decomposition approach.

*3.4. Computing UB.* To compute the UB, one way is to first remove constraints (3). Solve the $P$ formulation for a LP solution. Then, use a technique called randomized rounding (RR) with a new technique called the unit cost technique below, to treat the fractional solutions of $y$ as a probability distribution and keep opening them randomly to get enough capacity. We keep updating this UB using the VA every 50 passes, until we meet one of the stopping criteria. We choose the passes to be 50 because the UB changes slowly, and from our experiments, 50 passes usually show some improvement in the UB. This is one way of obtaining the UB.

The other way is similar to the above, except after we solve the $P$ problem; we call the upper bound of the greedy weak representation. We call this strategy the "warm start UB." This UB results in a smaller step size of the VA. Therefore, both the LB and the UB have converged faster in our experiments. Then, after 50 passes, we call the RR technique below as before. We keep using the VA and the RR technique until we meet one of the stopping criteria.

A third technique called the "unit cost technique" (UC) selects the $y$ı s to open according to their costs. It chooses the ones with the minimum unit cost. We call this technique once at the beginning before calling the RR. Then after 50 passes, we switch to call the RR and keep calling the RR every 50 passes to update the UB until one of our stopping criteria is met. Next, we discuss these techniques in more detail.

*3.5. Discussion.* Combining the approaches discussed above, which were used to obtain the LB and the UB, we got eight different approaches to solve the CFLP. Figure 1 below is a tree chart to simplify these eight approaches that are denoted by methods LR1–LR8.

These methods are used in solving different sizes of the CFLP. To decide which methods are used to solve which problem, we classify the problems into three classes: the small, the medium, and the large.

## 4. Computational Results

The selected model collection was drawn from Beasley [14], and we selected two instances where they fail to find feasible solutions when they use the factor = 1.5. The factor value is defined as $\sum_{i \in F} S_i / \sum_{j \in D} d_j$, which is used to rescale the capacity values. They used heuristics that produce a feasible integer solution and used a Lagrangian relaxation and the volume algorithm to obtain a lower bound on the optimal value. For the CFLP, they were able to solve instances up to $1000 \times 1000$.

We use instances generated as in the work of Beasley [7] and the work of Ravi and Sinha [9], which are as follows:

(i) We generate the demand points $j$ and the facility points $i$ uniformly at random in a $[0, 1] \times [0, 1]$

(ii) The demand values $d_j$ are generated from a uniform distribution with mean 35 and standard deviation 5, $U[35 - \sqrt{15}, 35 + \sqrt{15}]$

(iii) The capacities $S_i$ are generated from a $U[10, 160]$ distribution and are then rescaled so that $\sum_{i \in F} S_i / \sum_{j \in D} d_j$ was set equal to the parameter factor; this was either set at 10 and in some cases 1.5

(iv) The fixed cost $f_i$ of each site was set using the formula $f_i = U[0, 90] + U[100, 110]\sqrt{S_i}$

(v) The unit transportation costs $c_{ij}$ correspond to the Euclidean distance scaled by 10

Next, we report the results of our experiments for two algorithms using a number of approaches denoted by methods LR1–LR8. The first are the approaches of the Lagrangian relaxation algorithm without decomposition (LR1–LR4), and we denote this algorithm by LR and VA. The second are the approaches of the Lagrangian relaxation with the decomposition algorithm (LR5–LR8), and we denote this algorithm by LD and VA.

Recall that, we classify the problems into three classes according to the size. The classes are small, medium, and large. In the next subsections, we report the results and the analysis of each class of the problems.

*4.1. The Small Size Problems.* In Table 1 below, we report the solutions for solving the small problems of size 100–300. By comparing the solutions of the two algorithms explained above, we notice that both the LB and the UB of the Lagrangian decomposition with the volume algorithm LR5–LR8 are much better in both quality and running time. There is a big difference in the LB between the two algorithms, and the reason is that one of the advantages of the LD algorithm is forcing $y'$s to be 1. In addition, we were able to

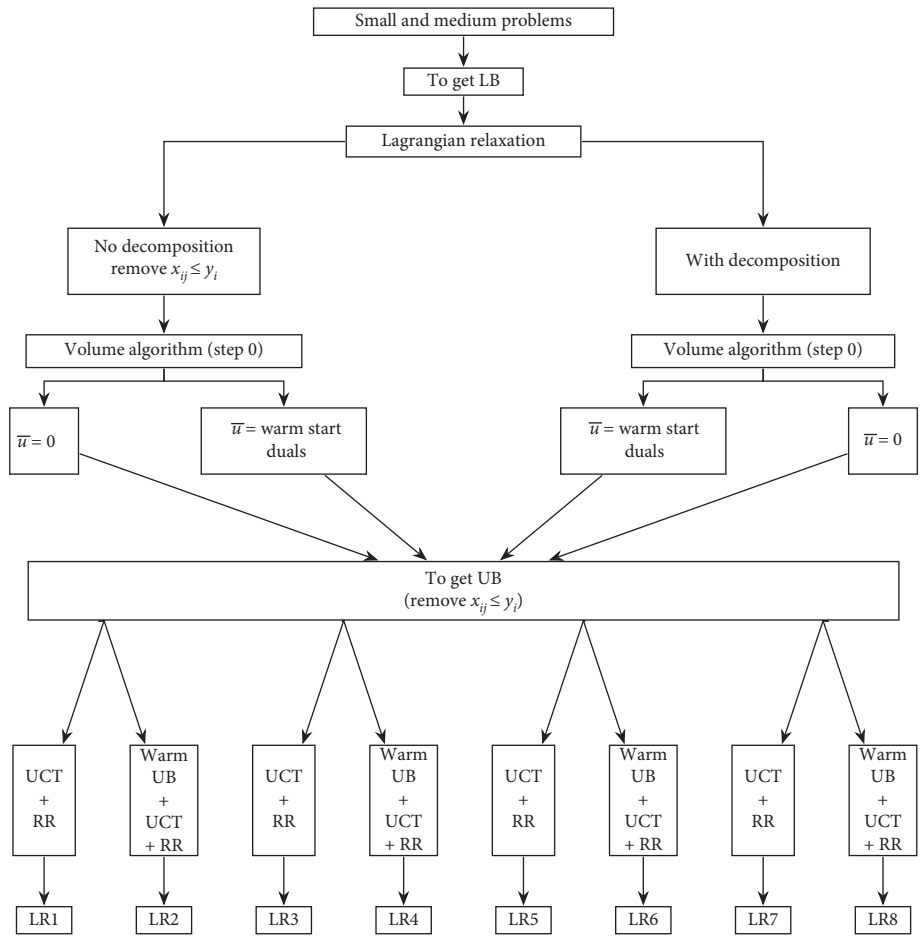


Figure 1: The Lagrangian heuristics and the volume algorithm's methods LR1–LR8.

Table 1: A comparison between the results of the LR and VA and the LD and VA for the small problems sized 100–300.

| Problem size $m \times n$ | Algorithm | Method | LB | UB | Time | Passes | Error % |
|---|---|---|---|---|---|---|---|
| $100 \times 100$ | LR and VA | LR1 | 9529.44 | 11984.03 | 64.48 | 650 | 20.5 |
| | | LR2 | 9529.44 | 11757.42 | 359.4 | 1750 | 18.9 |
| | | LR3 | 9529.44 | 11433.22 | 113.8 | 571 | 16.7 |
| | | LR4 | 9529.44 | 11433.22 | 41.0 | 423 | 16.7 |
| | LD and VA | LR5 | 11088.854 | 11717.01 | 4.35 | 450 | 5.43 |
| | | LR6 | 11088.854 | 11599.43 | 3.8 | 397 | 4.46 |
| | | LR7 | 11088.854 | 11433.22 | 4.61 | 482 | 3.03 |
| | | LR8 | 11088.854 | 11433.22 | 4.12 | 443 | 3.02 |
| | Best greedy sol. | Filter 4 | 11089.6 | 11634.30 | 55.19 | — | 4.68 |
| $200 \times 200$ | LR and VA | LR1 | 18005.55 | 20677.407 | 681.68 | 1400 | 12.9 |
| | | LR2 | 18005.75 | 20447.484 | 680.48 | 1450 | 11.9 |
| | | LR3 | 18005.34 | 20083.69 | 822.64 | 617 | 10.3 |
| | | LR4 | 18005.75 | 20083.69 | 78.10 | 503 | 10.3 |
| | LD and VA | LR5 | 19542.11 | 19919.04 | 151.54 | 1850 | 1.89 |
| | | LR6 | 19542.02 | 19919.05 | 75.06 | 900 | 1.89 |
| | | LR7 | 19542.08 | 19919.05 | 116.26 | 1400 | 1.89 |
| | | LR8 | 19542.07 | 19792.52 | 95.01 | 1150 | 1.30 |
| | Best greedy sol. | Filter 4 | 19542.3 | 19791.608 | 791.64 | — | 1.30 |
| $300 \times 300$ | LR and VA | LR1 | 26428.89 | 30775.85 | 1552.28 | 1000 | 14.0 |
| | | LR2 | 26428.09 | 29854.86 | 774.10 | 500 | 11.5 |
| | | LR3 | 26428.85 | 29092.27 | 1000.82 | 647 | 9.2 |
| | | LR4 | 26428.86 | 29092.27 | 644.98 | 438 | 9.2 |
| | LD and VA | LR5 | 28351.01 | 29197.79 | 616.77 | 702 | 2.9 |
| | | LR6 | 28351.01 | 29065.70 | 1469.08 | 1700 | 2.5 |
| | | LR7 | 28351.01 | 28933.05 | 935.91 | 1150 | 2.01 |
| | | LR8 | 28351.01 | 28933.05 | 369.46 | 1150 | 2.01 |
| | Best greedy sol. | Filter 4 | 28351.8 | 29125.771 | 5403.93 | — | 2.7 |

Table 2: Computational results of the LD and VA for the medium problems of size 400–900.

| Problem size $m \times n$ | Algorithm | Method | LB | UB | Time | Passes | Error % |
|---|---|---|---|---|---|---|---|
| $400 \times 400$ | LR and VA | LR5 | 36935.02 | 37896.53 | 1069.32 | 1350 | 2.50 |
| | | LR6 | 36935.06 | 37490.2 | 1444.27 | 1800 | 1.50 |
| | | LR7 | 36935.04 | 37822.04 | 487.29 | 625 | 2.34 |
| | | LR8 | 36935.42 | 37822.0 | 451.83 | 552 | 2.34 |
| | Best greedy sol. | Mweak | 34834.849 | 37495.195 | 431.67 | — | 7.1 |
| $500 \times 500$ | LR and VA | LR5 | 45409.13 | 45785.48 | 1212.02 | 618 | 0.82 |
| | | LR6 | 45410.19 | 45732.0 | 6285.19 | 1550 | 0.70 |
| | | LR7 | 45409.29 | 45644.58 | 1902.88 | 1000 | 0.50 |
| | | LR8 | 45409.0 | 45694.22 | 2158.55 | 1050 | 0.62 |
| | Best greedy sol. | Mweak | 43217.95 | 45968.784 | 864.89 | — | 6.0 |
| $600 \times 600$ | LR and VA | LR5 | 53756.957 | 54788.362 | 4572.55 | 901 | 1.90 |
| | | LR6 | 53709.728 | 54535.212 | 3333.43 | 651 | 1.50 |
| | | LR7 | 53756.414 | 54285.848 | 4133.19 | 601 | 0.96 |
| | | LR8 | 53687.616 | 54285.848 | 3135.78 | 601 | 0.98 |
| | Best greedy sol. | Mweak | 51564.439 | 54065.88 | 1541.69 | — | 4.60 |
| $700 \times 700$ | LR and VA | LR5 | 61165.793 | 62260.506 | 5553.19 | 601 | 1.76 |
| | | LR6 | 61166.521 | 62230.049 | 6574.87 | 751 | 1.71 |
| | | LR7 | 61165.194 | 62048.79 | 5058.79 | 601 | 1.40 |
| | | LR8 | 61165.867 | 62048.793 | 4972.56 | 601 | 1.4 |
| | Best greedy sol. | Mweak | 58997.209 | 61710.99 | 2193.92 | — | 4.4 |
| $800 \times 800$ | LR and VA | LR5 | 68783.862 | 69557.30 | 12192.85 | 1100 | 1.1 |
| | | LR6 | 68724.139 | 69956.659 | 13840.73 | 1001 | 1.76 |
| | | LR7 | 68764.846 | 69049.99 | 8601.68 | 602 | 0.413 |
| | | LR8 | 68744.645 | 69049.99 | 8582.22 | 601 | 0.44 |
| | Best greedy sol. | Weak | 66693.20 | 69049.99 | 543.63 | — | 3.4 |
| $900 \times 900$ | LR and VA | LR5 | 76509.406 | 77139.556 | 12311.42 | 604 | 0.82 |
| | | LR6 | 76421.149 | 77028.705 | 12156.0 | 603 | 0.79 |
| | | LR7 | 76511.347 | 77028.705 | 9715.03 | 601 | 0.67 |
| | | LR8 | 76505.026 | 77028.705 | 9723.46 | 602 | 0.68 |
| | Best greedy sol. | Weak | 74423.0 | 77054.107 | 1029.58 | — | 3.4 |

Table 3: Computational results of the LD and VA for the large problems sized 1000–3000.

| Problem size $m \times n$ | Algorithm | LB | UB | Time | Passes | Error % |
|---|---|---|---|---|---|---|
| $1000 \times 1000$ | LD and VA | 84448.103 | 86732.678 | 1140.030 | 1550 | 2.630 |
| | Greedy weak | 82466.800 | 87710.265 | 1121.890 | — | 5.980 |
| $1200 \times 1200$ | LD and VA | 101617.237 | 103125.198 | 1705.680 | 882 | 1.460 |
| | Greedy weak | 98770.100 | 103579.320 | 1766.210 | — | 4.640 |
| $1500 \times 1500$ | LD and VA | 124282.603 | 126124.230 | 2429.820 | 1100 | 1.460 |
| | Greedy weak | — | — | — | — | — |
| $2000 \times 2000$ | LD and VA | 161914.359 | 164486.627 | 9100.300 | 1760 | 1.564 |
| | Greedy weak | — | — | — | — | — |
| $2500 \times 2500$ | LD and VA | 202179.251 | 202784.372 | 21155.3 | 1950 | 0.300 |
| | Greedy weak | — | — | — | — | — |
| $3000 \times 3000$ | LD and VA | 239858.612 | 241723.867 | 38016.52 | 1993 | 0.772 |
| | Greedy weak | — | — | — | — | — |

obtain a closed LB as greedy's of the strong representation of the problem, but a better UB and running time. Also, the error values are less than greedy's in some of our methods LR5–LR8.

In the next tables, the column "passes" presents the number of times we go through the algorithm looking for a better LB and UB. The last row corresponds to the best greedy solution we were able to obtain; we include it in these tables for comparison purposes. The last column corresponds to the error percentage values, which can be obtained by (UB − LB)/UB. This value shows the percentage gap between the lower and the upper bounds of the problem. We use this value as a stopping criterion if it is less than 2%. Also, the stopping criteria here in most of the results are the number of passes, and in the rest, they are the error values.

*4.2. The Medium Size Problems.* Table 2 below lists all the results of solving the medium problems sized 400–900. We used the Lagrangian decomposition algorithm only because the size of the problem is too large to solve without decomposing them. The results of problem sized $500 \times 500$ shows that we were able to obtain a better LB and UB compared to those of greedy's solution. Also, the gap between them is less than the gap of greedy's solution. In problems sized 500–900, we were able to obtain a feasible integer solution (UB) within a gap of less than 1% of the lower bound LB in most of our solutions and in others less than 2.5%. The stopping criterion was the error value in most of the results.

*4.3. The Large Size Problems.* In Table 3, we illustrate all the results of solving large-sized problems using the Lagrangian decomposition with the volume algorithm, which is the goal of this research. We were able to solve large instances of the CFLP. The largest problem solved was of size $3000 \times 3000$. In some of the problems, we were able to compare the solution with the greedy's solution. Table 3 shows that we obtained a better LB and UB than the greedy's. Also, the gap between them is much less than the gap between the UB and the LB of the greedy's. In all of the problem solution, the gap is less than 2.63%.

## 5. Conclusion

In this article, we have investigated solving very large instances of the CFLP. We first presented a general algebraic model for the CFLP. In addition, the created Lagrangian decomposition representation solves large instances of the CFLP. We have improved the lower bound of this technique by both exploiting the decomposition approach as well as introducing a tightening constraint, which is a new positive addition to the Lagrangian decomposition approach for solving the CFLP.

In the case of very large problems, we have applied a randomized rounding technique along with our unit cost technique, which provided a good UB in a reasonable computational time.

The thrust of this paper research was to be able to solve very large instances of the CFLP. We have illustrated that our algorithm scales up. Our results show that increasing the problem size leads to a small relative error between the LB and the UB without too much burden on the computational time. In order to process such large model instances, we have exploited a sparse technology technique in implementing our algorithm. Thus, we have developed a sparse data structure that we exploit in our algorithm implementation.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The author declares that there are no conflicts of interest.

## References

[1] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, 1970.

[2] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees: part II," *Mathematical Programming*, vol. 1, no. 1, pp. 6–25, 1971.

[3] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, no. 1, pp. 62–88, 1974.

[4] F. Barahona and R. Anbil, "The volume algorithm: producing primal solutions with a subgradient method," *Mathematical Programming*, vol. 87, no. 3, pp. 385–399, 2000.

[5] K. Aardal, "Capacitated facility location: separation algorithms and computational experience," *Mathematical Programming*, vol. 81, pp. 149–175, 1998.

[6] K. Aardal, "Reformulation of capacitated facility location problems: how redundant information can help," *Annals of Operations Research*, vol. 82, pp. 289–308, 1998.

[7] 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.

[8] J. E. Beasley, "Lagrangean heuristics for location problems," *European Journal of Operational Research*, vol. 65, no. 3, pp. 383–399, 1993.

[9] R. Ravi and A. Sinha, "Approximation algorithms for problems combining facility location and network design," *Operations Research*, vol. 54, no. 1, pp. 73–81, 2006.

[10] M. T. Hajiaghavi, M. Mahdian, and V. S. Mirrokni, "The facility location problem with general cost function," *Networks*, vol. 42, no. 1, pp. 42–47, 2002.

[11] M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani, "A greedy facility location algorithm analyzed using dual fitting," *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, Springer- Verlag LNCS, vol. 2129, pp. 127–137, Berlin, Germany, 2001.

[12] E. J. Alenezy and R. F. Khalaf, "Implementing lagrangean decomposition technique to acquire an adequate lower boundon the facility location problem solution," *Applied Mathematics*, vol. 4, no. 8, pp. 1168–1172, 2013.

[13] D. Serra, S. Ratick, and C. Revelle, "The Maximum Capture problem with uncertainty," *Environment and Planning B: Planning and Design*, vol. 23, no. 1, pp. 49–59, 1996.

[14] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *The Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.