ISSN 1748-7595 (Online)

Working Paper Series

A Guided Reactive GRASP for the Capacitated Multi-Source Weber Problem*

Martino Luis
College of Business, Universiti Utara Malaysia,

Said Salhi
Kent Business School

Gabor Nagy
Kent Business School

***This research has been partially supported by the Ministry of Science and Technology of Spain under the research project ECO2008-00667/ECON, in part financed by the European Regional Development Fund (ERDF).**

Working Paper No. 236
December 2010

A Guided Reactive GRASP for the Capacitated Multi-Source Weber Problem*

Martino Luis[a], Said Salhi[b] and Gábor Nagy[b]

[a]College of Business, Universiti Utara Malaysia,
06010 UUM Sintok, Kedah Darul Aman, Malaysia,
Email: martino@uum.edu.my

[b]The Centre for Logistics & Heuristic Optimisation, Kent Business School,
University of Kent at Canterbury, Canterbury CT2 7PE, UK
Email: {S.Salhi, G.Nagy}@kent.ac.uk

Abstract

The capacitated multi-source Weber problem entails finding both the locations of capacitated facilities on a plane and their customer allocations. A framework that uses adaptive learning and functional representation to construct the restricted candidate list (RCL) within a greedy randomized adaptive search procedure (GRASP) is put forward. An implementation of restricted regions that forbids new facilities to be located too close to the previously found facilities is also embedded into the search to build up the RCL more efficiently. The performance of this GRASP based approach is tested on three classes of instances with constant and variable capacities. Very competitive results are obtained when compared to the best known results from the literature.

Keywords: Reactive GRASP, learning, continuous location, capacitated location, restricted regions.

1. Introduction

In this study, we are given a set of customers, located at *n* fixed points, with their respective demands, and we are required to locate *M* facilities in the plane to serve these *n* customers, and to find the allocation of these customers to these *M* facilities without violating the capacity of any of the facilities. It is usually assumed that all facilities have the same given capacity *b*. However, we will also look at the case where the facilities may take different capacities, see section 6. The objective is to minimize the sum of the weighted Euclidean distances. This capacitated continuous location-allocation problem is also known as the capacitated multi-source Weber problem (CMSWP) and can be formulated as follows:

$$Minimize \sum_{i=1}^{M}\sum_{j=1}^{n} x_{ij} d(X_i, a_j) \qquad (1)$$

Subject to

$$\sum_{i=1}^{M} x_{ij} = w_j, \quad j = 1, \ldots, n \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} \le b, \quad i = 1, \ldots M \qquad (3)$$

$$X_i = (X_i^1, X_i^2) \in \Re^2 ; x_{ij} \ge 0, \qquad i = 1, \ldots, M; j = 1, \ldots, n \qquad (4)$$

where

M : an upper bound on the number of facilities to be located;

x_{ij} : the quantity assigned from facility i to customer j, $i = 1, \ldots M$; $j = 1, \ldots, n$;

$d(X_i, a_j)$: the distance between facility i and customer j;

$a_j = (a_j^1, a_j^2) \in \Re^2$ is the given location of customer j;

$X_i = (X_i^1, X_i^2) \in \Re^2$ are the coordinates of facility i;

w_j : the demand, or weight, of customer j, where $w_j \in \mathcal{N}$;

b : a fixed capacity of a facility.

The objective function (1) is to minimize the sum of the transportation costs. Constraints (2) guarantee that the total demand of each customer is satisfied. Constraints (3) ensure that the capacity constraints of the facilities are not exceeded, and constraints (4) refer to the continuous location variables as well as the non-negativity of the allocation variables. Note that once the M facilities are located, the problem reduces to the classical Transportation Problem (TP). We consider the problem to have at least one feasible solution by assuming that $\sum_{j=1}^{n} w_j \le Mb$. The case when there is a number of capacities available from which one chooses the sizes of the facilities will be briefly described in Section 6.

The optimal solution of this continuous location problem could be infeasible in practice as the locations could be on lakes, mountains, etc. Nonetheless there are applications for the MSWP such as the location of oil drills in the sea or the desert. For instance, Rosing (1992) finds the location of steam generators in the Orinoco heavy oil belt of Venezuela by solving the MSWP. Furthermore, a solution of this problem could also be used as a lower bound for the discrete problem (see Daskin, 2008), as a way of identifying additional potential sites in the neighbourhood of these continuous locations, or it can simply be transformed into a feasible solution that requires the presence of barriers for example see Canbolat and Wesolowsky (2010). The problem is NP hard as it can be transformed into either of the following problems. The first is the Multi-source Weber problem, shown to be NP hard by Megiddo and Supowit (1984), which can be obtained by relaxing the capacity constraints (3). The second is the capacitated discrete location problem, also known to be NP hard (see Mirchandani and Francis (1990)), which can be derived by restricting the continuous space to the discrete space of the potential sites only.

The remainder of this paper is organized as follows: in the next section we present a review of the relevant literature. The following two sections thereafter address our general framework of the GRASP heuristic. Section 6 describes the case when there are several capacities available to choose from, followed by a section on computational experiments. Finally, the last section summarizes our conclusions and outlines some research avenues that we believe to be worthwhile investigating in the future.

2. Literature Review

Most of the literature on the continuous location problem focuses on the (uncapacitated) multi-source Weber problem (MSWP). The solution approaches for this problem are mainly heuristic. Hansen *et al.* (1998) tackle the MSWP by solving the *p*-median problem exactly while considering all fixed points as potential facility sites and then apply the Weiszfeld equations to derive the continuous location. Brimberg *et al.* (2000) carry out a comparison of heuristics including those based on variable neighbourhood search and genetic algorithms. Gamal and Salhi (2001) present a constructive heuristic based on the furthest distance rule to find initial locations while introducing forbidden regions to avoid locations being too close to each other. Gamal and Salhi (2003) create a discretisation-based approach known as a cellular heuristic whereas Salhi and Gamal (2003) adopt a genetic algorithm to solve the problem. Taillard (2003) proposes a decomposition/recombination heuristic that partitions the problem into smaller subproblems which are then solved by a candidate list search for a large number of centres. Aras *et al.* (2006) tackle the MSWP using self-organizing maps for both Euclidean and rectilinear distances. A variant of this problem is the constrained Weber problem which is also known as the Weber problem in the

Presence of Forbidden Regions and/or Barriers to Travel. This was originally investigated by Katz and Cooper (1981) but recent studies on this particular topic can be found in Bischoff and Klamroth (2007), and Canbolat and Wesolowsky (2010) where interesting analytical results were derived. For more details and references on the MSWP the reader is referred to the comprehensive survey by Brimberg *et al.* (2008).

There is however a shortage of papers on the capacitated version of the MSWP (CMSWP for short). The earliest work in this area was conducted by Cooper (1972) who develops exact and heuristic methods. In the exact method, the idea is to generate all feasible basic solutions using the transportation problem. Starting with any feasible basic solution, the connected graph of all basic feasible solutions is constructed. For every solution, the location problem is solved and the solution which yields the minimum cost is chosen as the optimal solution. In the heuristic method, the alternating transportation-location heuristic known for short as ATL was proposed. Fundamentally, ATL is a modification of the heuristic method originally developed by Cooper (1964) for the pure location-allocation problem known as the alternate location-allocation method. The idea of ATL is that the location-allocation problem and the transportation problem (TP) are alternately solved until the improvement in the total cost becomes negligible. Cooper (1976) modifies his heuristic method (Cooper, 1975) initially used for the fixed charge problem. Sherali and Shetty (1977) solve the rectilinear distance CMSWP using an exact method known as a convergent cutting plane algorithm. This was originally derived from a bilinear programming problem by substituting decision variables by the difference of two non-negative variables.

It is only in 1992 that the problem was revisited by Sherali and Tuncbilek (1992). Their problem version, which uses a distance proportional to the square of the Euclidean, is transformed into a quadratic convex maximization problem. The authors propose an exact method based on a branch and bound algorithm to compute strong upper bounds via a Lagrangean relaxation scheme and a partitioning approach based on dichotomy that adopts a special structure of transportation constraints. Experiments were conducted using randomly generated data with the number of facilities (M) vary from 5 to 20. Sherali *et al.* (1994) formulate the rectilinear distance CMSWP as a mixed integer nonlinear programming formulation and proposed a reformulation-linearization technique (RLT) to transform the problem into a linear mixed-integer program. Gong *et al.* (1997) put forward a hybrid evolutionary method based on a genetic algorithm to search the locatable area and hence find the global or near global locations. In the allocation stage, a Lagrange relaxation method was applied. Experiments were carried out on randomly generated data with the number of facilities (M) varying from 2 to 6. Sherali *et al.* (2002) design a branch and bound approach based on a partitioning of the allocation space to develop global optimization procedures for the capacitated Euclidean and ℓ_p distances MSWP. Two bounding schemes were also put forward based on solving a projected location space subproblem and a variant of RLT that reformulated the problem into a linear programming relaxation. Aras *et al.* (2007a) propose three heuristic methods that use Lagrangean heuristic, the discrete p-capacitated facility location heuristic which is similar to the p-median method of Hansen *et al.* (1998), and the cellular heuristic of Gamal and Salhi (2003) to deal with the CMSWP with Euclidean, squared Euclidean, and ℓ_p distances. In a subsequent research, Aras *et al.* (2008) adopt their earlier approaches to solve the CMSWP with rectilinear distance. Aras *et al.* (2007b) tackle the CMSWP with rectilinear, Euclidean, squared Euclidean, and ℓ_p distances by adopting simulated annealing, threshold accepting, and genetic algorithms. These heuristics perform well when tested on the Sherali *et al.* (2002) data sets which are small sized instances ($n \leq 30$) where the capacity of the facilities is not necessarily constant. Zainuddin and Salhi (2007) present a perturbation-based heuristic for solving the Euclidean CMSWP. This heuristic outperformed the classical ATL when tested on large instances (n = 50 to 1060 and M =5 to 50) with facilities having equal capacity. Very recently, Luis *et al.* (2009) tackle the CMSWP by introducing the concept of region-rejection that restricts choosing areas which are too close to the previously selected locations. A discretisation technique of converting a plane into a discrete space is also presented. Competitive results are obtained when tested on previously published results.

3. Solution Framework

Our proposed solution method is based on Cooper's Alternating Transportation-Location Heuristic (see Cooper, 1972) which is integrated into a GRASP framework.

3.1. Cooper's Alternating Transportation-Location Heuristic

Initially, M open facilities are chosen from the customer points (fixed points), then the TP, using these M open facilities, is solved. The output is the M independent set of allocations, each subset consisting of n_i fixed points where i = 1, 2, …, M and $\sum_{i=1}^{M} n_i \geq n$. Note that we used '≥' instead of '=' as some customers may have their demand split between different facilities during the allocation stage. An iterative procedure based on the Weiszfeld equations, as given in Equation (5), is then applied to find the new location of each of the M facilities (i = 1, 2, …, M) .

$$X_i^{1(k)} = \frac{\sum_{j_i=1}^{n_i} \frac{w_{j_i} a_{j_i}^1}{d(X_i^{(k-1)}, a_{j_i})}}{\sum_{j_i=1}^{n_i} \frac{w_{j_i}}{d(X_i^{(k-1)}, a_{j_i})}} \quad \text{and} \quad X_i^{2(k)} = \frac{\sum_{j_i=1}^{n_i} \frac{w_{j_i} a_{j_i}^2}{d(X_i^{(k-1)}, a_{j_i})}}{\sum_{j_i=1}^{n_i} \frac{w_{j_i}}{d(X_i^{(k-1)}, a_{j_i})}} \tag{5}$$

where

the superscript k is the iteration number within the Weiszfeld iterative procedure;

$(a_{j_i}^1, a_{j_i}^2)$ represents the location of the j_ith fixed points where $j_i = 1, 2, \ldots, n_i$;

$(X_i^{1(k)}, X_i^{2(k)})$ denotes the new location of the ith facility at iteration k ($i = 1, 2, \ldots, M$) and

w_{j_i} corresponds to all or a fraction of the jth customer demand that is served by facility i.

In other words, the demand of some customers might have been split as a result of the solution of the TP (i.e., $w_{j_i} \le w_j$). Note that some customers may be utilized more than once in Equation (5) but at each time a portion w_{j_i} of their demand is used only. The process of alternating between the location-allocation problem and the transportation problem is then applied until no improvement in total cost can be found. Figure 1, which is used as a basis for our research, shows the main steps of Cooper's Alternating Transportation-Location Heuristic.

Step 1 Find the initial facility configuration, say $(X_i)_{i=1,..,M}$ and set a threshold value (tolerance) $\varepsilon = 0.001$

Step 2 Solve the TP using the initial facility configuration $(X_i)_{i=1,..,M}$ to obtain the allocation for the capacitated problem. Let A_i^{old} *be* the set of customers served by facility i and set the total cost $TCOST = \sum_{i=1}^{M} \sum_{j \in A_i^{old}} d(X_i, a_j)$

Step 3 Find the new location $(X_i^{new})_{i=1,..,M}$ using Equation (5) starting from $(X_i)_{i=1,..,M}$.

Step 4 Solve the TP using the new locations $(X_i^{new})_{i=1,..,M}$ to find the new corresponding allocation A_i^{new} and its total cost $TCOST^{new} = \sum_{i=1}^{M} \sum_{j \in A_i^{new}} d(X_i, a_j)$.

Step 5 If $|TCOST^{new} - TCOST| \le \varepsilon$ stop and choose the best facility configuration as $(X_i^* = X_i^{new})_{i=1,..,M}$ and the least cost as $TCOST^* = TCOST^{new}$,
Else set $(X_i)_{i=1,..,M} = (X_i^{new})_{i=1,..,M}$, $TCOST = TCOST^{new}$ and go to step 3.

Figure 1. Cooper's Alternating Transportation-Location Heuristic (ATL)

In Step 1 of Figure 1, Cooper (1972) selects locations randomly from customer locations as the initial facility configuration. In this study, we propose instead a novel reactive GRASP heuristic where we adopt the strengths of this approach in the construction of the initial solution while the local search we use is the standard application of the Weiszfeld equations as given in (5). This local search is an iterative pure descent method which can be shown to yield (or converge to) a local minimum. Though each facility, within its subset of assigned customers, converges to its global minimum through Equation (5) as proved by Kuhn (1973) for the case of the Euclidean distance, the overall solution concerning the location of all the facilities which is obtained by Cooper's ALT heuristic (see Figure 1), can be guaranteed to be a local minimum only. This theoretical convergence result is generalised for other distances as proved by Brimberg and Love (1993). Some may argue that this descent method namely the ATL of Figure 1, which happens to be appropriate for the continuous case, is not a standard local search which is usually and specifically defined by its move, its neighbourhood and its function evaluation. Of course one can follow these three steps by constructing a circle centred at each facility location with an appropriate radius to act as the neighbourhood with an infinite number of points, the move can be based on the generation of a random point inside each of the circles to replace the existing facility and finally the evaluation function is then just the computation of the new total distance between the generated points and their corresponding assigned customers. Note that this way, though applicable for the discrete case, is crude and inefficient for the continuous space as the ATL as described in Figure 1 is known to be an efficient iterative approach in producing a local minimum. Obviously one may also opt to improve this initial solution in the discrete space first through one of the standard local searches, say by defining a move such as exchanging customers between facilities, opening a new facility and dropping an existing one, etc. The obtained solution, which happens to be a solution for the *p*-median problem, can be used as the initial solution in the ATL procedure of Figure1 to yield a better location in the continuous space. Note that in Figure 1, the quality of the local solution is constrained by a certain tolerance specified by the user in Step 5 (here we set $\varepsilon = 0.001$). Schemes on how to obtain such initial solutions are presented in subsequent sections following a short description of the GRASP meta-heuristic.

3.2. A Brief Outline of GRASP

Greedy randomized adaptive search procedure (GRASP for short) is a two phase meta-heuristic method based on a multi-start randomized search technique to solve hard combinatorial optimisation problems. GRASP was initially studied by Hart and Shogan (1987) and then by Feo and Resende (1989) as a semi greedy heuristic. It was then formally introduced by Feo and Resende

(1995). Comprehensive reviews including some applications on this topic are given by Pitsoulis and Resende (2002), and Resende and Ribeiro (2003). For instance, an efficient implementation of GRASP for the case of the single source capacitated location problem is given by Delmaire *et al.* (1999). Recent related location problems that were investigated using GRASP methodology include the capacitated clustering by Deng and Bard (2010) who integrate GRASP with path relinking and the territory design problem by Rios-Mercado and Fernandez (2009) who put forward a powerful reactive GRASP.

The first phase of GRASP is a construction phase where a feasible initial solution is built one at a time. The construction of these feasible solutions is based on the creation of a restricted candidate list (RCL) made up of good attributes (facilities) including those facilities of the configuration that yields the best solution. From this set, one element is chosen one at a time either randomly or following a certain selection rule until the full solution is completed. The second phase is a standard local search used to explore the neighbourhood of the constructed solution in order to find a better solution. The two phases are reiterated several times either independently or using a certain learning scheme and the best overall local optimum is then selected as the final result. The main steps of the basic GRASP, which are adopted for the case of the continuous location problem, are given in Figure 2. For further reading and references, the reader will find the paper by Resende and Ribeiro (2005) to be useful.

```
Step 0  Set MAXRUNS to the maximum number of runs and define the local search with its
        corresponding neighbourhood(s) made up of an infinite number of points, say $N(.) \subset \Re^{2M}$ with
        $M$ denoting the number of facilities, and its function evaluation $F(.)$ representing the corresponding
        total transportation cost.
Step 1  (Initial Phase): Produce an initial solution for the $M$ discrete location problem, say $X_{cur}$ using a
        greedy-randomised procedure based on the RCL. Record the total transportation cost $F(X_{cur})$. In
        the first run set $X_{best} = X_{cur}$ and $F_{best} = F(X_{cur})$.
Step 2  (Improvement Phase/local search): Starting from $X_{cur}$, find the local minimum $X' \in N(X_{cur})$
        using Steps 3-5 of Cooper's ATL heuristic given in Figure 1.
Step 3  (Termination Phase): If $F(X') < F(X_{best})$ set $F_{best} = F(X')$ and $X_{best} = X'$.
        If the total number of runs reaches MAXRUNS the search stops, otherwise go to Step 1 to generate
        another initial solution either randomly or adaptively.
```

Figure 2. A Basic GRASP for the capacitated multi-source Weber problem

In our study, Step 1 will be implemented as if the problem is a discrete capacitated location problem with all customers acting as potential sites with M denoting the number of facilities to be located. The local search (Step 2 of Figure 2) is carried out using the iterative procedure made up of Steps 3 to 5 of Cooper's ATL as given in Figure 1. In the uncapacitated case, such a local search was shown to be promising in transforming a solution of a discrete location problem such as the *p*-median problem (an initial solution) into a solution in the continuous space such as the multi-source Weber problem (an improved solution) by Hansen *et al*. (1998) and Gamal and Salhi (2001).

In the remainder we will concentrate on exploring ways of defining the RCL and how to select an attribute from this subset using concepts borrowed from adaptive search.

3.3. The Construction of the RCL

GRASP requires a greediness function for selecting the new attribute to be chosen. In this study, we adopt an ADD procedure to be our greedy function. The idea behind the ADD heuristic is to add facilities one at a time until a required number of facilities, say M facilities, has been reached. Kuehn and Hamburger (1963) were among the first ones to adopt this scheme. For our local search phase, we use the Weiszfeld equations to improve the initial solution. Local search can also be used at the discrete stage before the Weiszfeld equations are applied to yield a continuous location solution.

In our implementation of the ADD scheme, all customer locations are considered as potential sites for being selected. The process of adding one facility at a time is carried out by adding temporarily a facility and then reallocating the customers to their nearest open facility (depot). The cost of the objective function is evaluated and the facility that yields the lowest cost is set as a starting point to make up the RCL. Given the excessive number of combinations, to reduce the computational burden, we consider a subset of customer locations only, say (S), instead of all customer locations. In addition, we also relax the restriction on the capacity of the facilities at this stage. The cardinality of this subset S is found empirically as follows:

$$|S| = \max\{20, \min(3M, 50)\} \qquad (6)$$

Let

$\boldsymbol{E}$: the current set of the selected facility locations, initially $\boldsymbol{E} = \{\varnothing\}$,

$\boldsymbol{A}_i$: the set of customers served by facility i ($i \in \boldsymbol{E}$).

$COST_l$: the incremental cost when facility l is temporarily added to $\boldsymbol{E}$ ($l \in S$) and is computed as

$COST_l = \sum_{i \in E} \vartheta_i$, $\forall l \in S$ with $\vartheta_i = \sum_{j \in A_i} d(X_i, a_j)$ (7)

The process of computing ϑ_i $\forall i \in \boldsymbol{E}$, when facility l is temporarily added, is given as follow:

```
For $i \in \boldsymbol{E}$-{$l$}
  For $j \in \boldsymbol{A}_i$,
    If $d(X_i, a_j) > d(X_l, a_j)$ {set $\boldsymbol{A}_i = \boldsymbol{A}_i$ - {$j$}; $\boldsymbol{A}_l = \boldsymbol{A}_l \cup$ {$j$}, $\vartheta_i = \vartheta_i - d(X_i, a_j)$ and $\vartheta_l = \vartheta_l + d(X_l, a_j)$ }
```

Let $g(l)$ = ($COST_l$) be the greedy function within GRASP used for the selection of the lth facility and define $g_{\max}$ and $g_{\min}$ as the maximum and the minimum cost for the function $g(l)$, $l \in S$.

There are two commonly used ways to build the RCL. The first one is based on the cardinality of the set as given by Hart and Shogan (1987) and the other is a value based approach where RCL is associated with a threshold parameter α as put forward by Feo and Resende (1989). The former uses a rank of the best k elements whereas in the latter candidates that have their greedy values, or cost, lying within α% of the best greedy function value, are chosen. In this research, we combine both designs when constructing our RCL. In other words, the proposed RCL consists of all elements $l \in S$ that have a function $g(l)$ value that lies between $g_{\min}$ ($\alpha = 0$) and $g_{\max}$ ($\alpha = 1$). This can be defined as

$$\text{RCL} = \{l \in S \,|\, g_{\min} \le g(l) \le g_{\min} + \alpha(g_{\max} - g_{\min})\} \text{ with } \alpha \in [0,1] \quad (8)$$

If $\alpha = 0$, the rule induced by (8) reduces to the greedy rule of selecting the element that produces the smallest cost whereas if α =1, (8) acts as a pure random search (i.e., all the elements will be candidates in RCL). The idea is to have a compromise between these two extremes while being closer to α =0. In addition, if RCL as found by (8) happens to be too small (i.e., |RCL| < $L_{\min}$), we relax the value of the parameter α by allowing the next ($L_{\min}$ - |RCL|) sorted elements in S, with respect to cost, to be added to the RCL. In this study we set $L_{\min} = \max(5, [M/2])$.

3.4. Selection of the Initial Solution (Facility Configuration)

At each iteration of the selection of the new facility (i.e., 1 to M), the attributes from the RCL (i.e., the potential facilities for being selected) are chosen pseudo-randomly based on their respective incremental cost $COST_l$. This selection procedure is repeated until M distinct facilities are generated.

A pseudo-code for the construction of the RCL and the selection of the facility configuration in Step 1 of GRASP is summarized in Figure 3. The complexity of this mechanism is of the order

($O(n^2 Log(n))$. This is performed as the choice of each facility (there are *M*) requiring $2S\log(S)$ operations for computing $g_{\min}$ and $g_{\max}$ for which the computation of the cost $g(l)$ needs *n* operations (*n* customers at most at each iteration). Overall this accounts to *2MnSLog(S)* which reduces to $O(n^2 Log(n))$ as *M* is a constant and *S* can go up to *n* in the worst case.

```
Step 0  Set E={$\varnothing$}, $|S| = \max(20, \min(3M, 50))$, $L_{\min} = \max(5, M/2)$ and $\alpha \in ]0,1[$.
Step 1  For all $l \in S$ compute $COST_l$ using Equation (7) and set $g(l) = COST_l$.
        Compute $g_{\min} = Min_{l \in S}(g(l))$ and $g_{\max} = Max_{l \in S}(g(l))$ and construct RCL using Equation (8).
Step 2  If |RCL| < $L_{\min}$ sort the elements of S in ascending order of the cost and select the top
        |$L_{\min}$-|RCL|) elements to complement the RCL.
Step 3  Choose facility $l^* \in RCL$ pseudo-randomly (based on cost) and set E = E $\cup$ {l*}.
Step 4  If |E| <M go to Step 1, else stop and take E as the set of facility configuration.
```

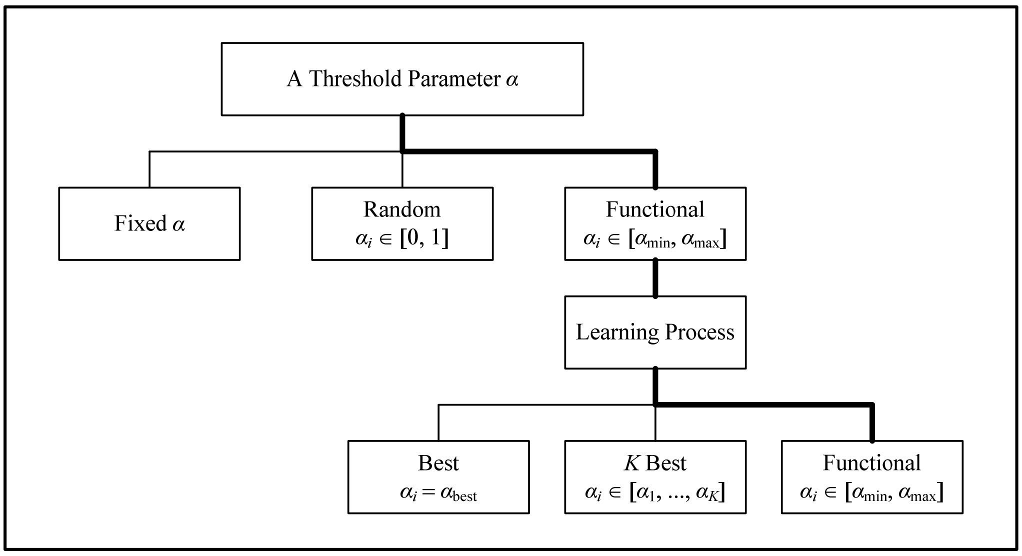
Figure 3. The construction of the RCL and the selection of the initial solution

4. Schemes for Generating α

In our first implementation of GRASP, we set the threshold parameter α to a predefined value. We then select α for each GRASP iteration randomly from the uniform distribution, *U*[0,1], and also following a functional-based scheme using a non-increasing linear function in the number of iterations. To make the search more adaptive, a learning phase that finds the best value of the parameter α and its appropriate range when used in the above schemes is also introduced. This framework is summarized in Figure 4 with 'bold lines' representing the path we are following in this paper. A description of these schemes is presented in the next subsections.

4.1 GRASP with Fixed α

The idea is to construct the RCL by using a fixed value of the threshold parameter α throughout the search. This is usually determined from preliminary experiments when several values are first tested and the most promising one is recorded. We refer to this GRASP implementation as GFAP for short. In this study we set the value of $\alpha = 0.2$. This figure was found to produce generally better results based on Equation (8) and some preliminary experiments when values of $\alpha = 0.1$ to 0.9 with a step size of 0.1 were used.



A Threshold Parameter α
Fixed α
Random $\alpha_i \in [0, 1]$
Functional $\alpha_i \in [\alpha_{\min}, \alpha_{\max}]$
Learning Process
Best $\alpha_i = \alpha_{\text{best}}$
K Best $\alpha_i \in [\alpha_1, ..., \alpha_K]$
Functional $\alpha_i \in [\alpha_{\min}, \alpha_{\max}]$

Figure 4. The Framework of Generating the Threshold Parameter α

4.2 GRASP with Random $\alpha \in U[0, 1]$

We aim to choose a new value for the parameter α in making up the RCL at each iteration as pointed out by Resende and Ribeiro (2003), instead of keeping it fixed throughout. One way is to generate randomly the value of α from the uniform distribution $U[0,1]$ at each GRASP iteration. We adopted this simple implementation in the construction of the RCL which we call GUAP for short.

4.3 A Functional-based α

This subsection discusses a mechanism to define the threshold parameter α within GRASP using a non-increasing function in the number of iterations. Here, we propose a decreasing linear function. Note that other functions such as concave and convex functions were also attempted but the results were found to be slightly inferior, see Luis (2008) for more details. This scheme starts by setting α_0 and $\alpha_{\min}$ which correspond to the initial and the final value of α, respectively. In other words, the value of α gradually decreases from α_0 to $\alpha_{\min}$ proportionally as follow:

$$\alpha_i = \alpha_0 - i\phi \text{ with } \phi = \frac{\alpha_0 - \alpha_{\min}}{M} \qquad \text{for } i = 1, \ldots, M$$

In this approach, we set the value of $\alpha_0 = 0.30$ and $\alpha_{\min} = 0.10$. These values are based on our preliminary experiments (see subsection 4.1) that showed that the best fixed value of α was found to be 0.2 which we then used as a centre location for our small range. The algorithm which uses this linear function is referred to as GALNP for short.

5. Guiding GRASP

In this section, we propose two approaches to guide GRASP and hence enhance its implementation. The first is to incorporate information through learning from the past whereas the second is to introduce some forbidden regions that are continuously constructed during the search to guide the search.

5.1 Enhancing GRASP via a Learning Process

The idea is to learn through gathering information from the previous iterations which are then used in subsequent iterations. As GALNP showed to outperform the other two variants (see Luis (2008) for details), we have based our learning scheme on this variant only. The reasoning behind this scheme is to find the range for α, (i.e., $[\alpha_{\min}, \alpha_0]$) after running GUAP K_0 times, say a proportion of the total number of runs (K). Subsequently, the obtained range for the parameter α is then used to continue the execution of the remaining runs of the GALNP. Here, we used a range size of 0.3 as this value was shown to provide good results in our preliminary testing. We constructed ranges of the form [0.1 - 0.4], [0.2 - 0.5], ..., [0.6 - 0.9] leading to 6 overlapping ranges. We recorded those results that are not too far away from the overall best and assigned their respective α values to the corresponding ranges (some may be counted more than once). The range which has the highest number of solutions is then chosen as our final range $[\alpha_{\min}, \alpha_0]$ to be used for the remaining ($K - K_0$) runs. We refer to this scheme with learning as GALNPL for short and its main steps are summarized in Figure 5.

Step1 Run the GUAP approach for K_0 runs (say $K/2$ where K refers to the maximum number of runs to be used),
Record $(\alpha_k)_{k=1,\ldots,K_0}$ (i.e. α for the k^{th} run), the corresponding facility configuration $(\sigma_k)_{(k=1,\ldots,K_0)}$ and cost $(Cost_k)_{(k=1,\ldots,K_0)}$.
Determine the least cost of these configurations as
$C^* = Min(Cost_k, k=1,\ldots,K_0)$

Step 2 Set the range size as $n_0 = 0.3$ and construct the following ranges as follows $[A_s, B_s]$ with $B_s = A_s + n_0 \ \ \forall s = 1,\ldots,6$; $A_1 = 0.1$ and $B_6 = 0.9$ leading to 6 overlapping ranges of width n_0 each. Set the minimum number of solutions for which a range will be appropriate for further selection as $N_{\min}$ = max (2, min (0.05 K_0, 5)) and set the correction factor $\varphi = 0.01$.

Step 3 Compute the subset of facility configurations that pass the threshold as
$G = \{(\sigma_k)_{(k=1,\ldots,K_0)} / Cost_k \le C^*(1+\varphi)\}$ and record their corresponding $(\alpha_k)_{k=1,\ldots,K_0}$ in the set $\Gamma = \{(\alpha_k)_{k=1,\ldots,K_0} / \sigma_k \in G\}$.
If there are no new attributes in *G* (initially $G = \{\varnothing\}$), set $\varphi = 2\varphi$ and repeat Step 3.

Step 4 In each range $[A_s, B_s]$; $s = 1,\ldots,6$ determine the number of solutions from G, say N_s.

Step 5 Choose the range $[A_{s^*}, B_{s^*}]$ for which $N_{s^*} = Max(N_s, s = 1,\ldots,6)$
If $N_{s^*} \ge N_{\min}$ stop and select the corresponding range $[A_{s^*}, B_{s^*}]$ as $[\alpha_{\min}, \alpha_0]$ that will be used in the remaining (K - K_0) runs in GALNP.
Else set $\varphi = 2\varphi$ and return to Step 3.

Figure 5. Step by Step Description of the GALNPL Approach

5.2 Guiding GRASP via the Presence of Restricted Regions

The idea is to introduce additional information into the search using restricted regions. This was originally and successfully implemented by Luis *et al.* (2009) to solve the CMSWP. Here, we adopt this idea to guide the construction of the RCL. In other words, the RCL will only contain promising elements which are outside the restricted regions at any given iteration of GRASP. We briefly discuss this concept here. The idea is to build restricted regions around the previously selected locations to avoid the new ones to be too close to the previous ones. A restricted region is defined by a circle whose radius can be either a predefined value based on the sparsity of the customers and the number of facilities or be dynamically adjusted using an iterative procedure. In this study, we only adopt a dynamic radius as the authors already showed (see Luis *et al.*, 2009) that the latter yields better results. The RCL, formed as in Equation (8), is then reconstructed by embedding these restricted regions. The condition of accepting or rejecting a candidate location, say *y*, is defined as follows:

$$d(y, X_i) > r_i, \forall X_i \in \boldsymbol{E} \qquad (9)$$

Where $\boldsymbol{E}$ denotes the set of previously selected facilities, and r_i is the radius of the circle whose centre is the location X_i. For more details about the way the r_i are defined and dynamically adjusted, one can refer to Luis *et al.* (2009).

Candidate locations are therefore included in the RCL if and only if elements in Equation (8) also satisfy Equation (9). As a result, the RCL with restricted regions, say $\hat{\text{RCL}}$, is defined as

$$\hat{\text{RCL}} = \{l \in S \,|\, g_{\min} \le \hat{g}(l) \le g_{\min} + \alpha_i (g_{\max} - g_{\min})\} \qquad (10)$$

where $\hat{g}(l)$ denotes the increment cost function when adding a facility l, $l \in S$, to the already chosen locations while satisfying (9).

The algorithm GALNP with the introduction of region rejections (RR) is implemented with and without learning which we refer to as GALNPRR and GALNPLRR for short respectively.

6. A simple adaptation to the case of several possible capacities

In certain situations, the capacity of a facility may not be fixed a priori and can take a set of available capacities. Here we introduced a simple but quick procedure to assign or match facilities to capacities. Every time the uncapacitated problem is solved, we record the total demand from each of the configurations and then use Vogel's Approximation method (VAM) (Bazaraa *et al*., 2005) to find an approximate assignment. Here, the costs for each row and column are represented by the absolute difference between the prospective supply and the total demand from each configuration. Once this cost matrix is constructed, VAM is applied to assign or match the supply/capacity to a facility. The process then continues as before. As some published results exist for small instances, the results for this case will be provided in the next section under Class II instances and then extended to large instances where no previous results exist in Class III instances. A mathematical formulation for this related problem which is similar to the one shown in the introduction section is given here as follows:

$$\textit{Minimize} \ \sum_{i=1}^{M}\sum_{j=1}^{n} x_{ij} d(X_i, a_j) + \sum_{k=1}^{T} F_k \sum_{i=1}^{M} z_{ik} \qquad (11)$$

Subject to

$$\sum_{i=1}^{M} x_{ij} = w_j, \quad j = 1, \ldots, n \qquad (12)$$

$$\sum_{j=1}^{n} x_{ij} \le \sum_{k=1}^{T} b_k z_{ik}, \quad i = 1, \ldots M \qquad (13)$$

$$\sum_{k=1}^{T} z_{ik} = 1 \qquad \forall i = 1,\dots,M \tag{14}$$

$$X_i = (X_i^1, X_i^2) \in \Re^2 \; ; x_{ij} \geq 0 \, , \;\; z_{ik} \in \{0,1\} \qquad i = 1, \dots, M; j = 1,\dots, n; k = 1,\dots, T \tag{15}$$

where

b_k: the k^{th} possible capacity, $k=1,..,T$ where T represents the number of possible capacities;

$F_k = kM$ with M a large positive penalty (this setting will guarantee that the use of the smallest possible capacity in the optimal solution);

$z_{ik} = 1$ if the k^{th} capacity is used for facility i; $i = 1, \dots M$; $k = 1, \dots, T$; and 0 otherwise.

The remaining notations are the same as described earlier in the introduction section.

The objective function (11) includes the transportation cost and the fixed facility cost. Constraints (12) show that every customer is fully served. Constraints (13) guarantee that one of the capacities will be able to accommodate the total allocation from a given facility *i*, and constraints (14) impose that there would be only one type of capacity used for each of the *M* facilities. Constraints (15) relate to the real, non negative and binary nature of the variables used.

In the implementation of our heuristic, the second term of (11) is not implicitly incorporated as we guarantee the use of the smallest capacity that is able to accommodate the total customer allocation from each facility. In addition, if two or more facilities have the same capacity, we choose randomly one of them.

7. Computational Results

The proposed methods were coded in C++, compiled with Salford FTN95 compiler, and run on a PC with an Intel 1.5 GHz Pentium M processor and 1.3 GB RAM. We tested our approaches on three classes of instances. In Class I, we adopted instances originally used for the MSWP (Brimberg *et al*., 2000). In Class II, we used the instances designed by Al-Loughani (1997) and Sherali *et al*. (2002). In Class III, we adapted Class I instances by incorporating various capacity values. Based on our experiments, the number of runs, *K*, is defined as follows:

$$K = \begin{cases} \max(100, 5M), & n \leq 50 \\ \max(100, M\sqrt[3]{n}), & otherwise \end{cases} \tag{16}$$

Also, within each run, the search terminates when there is no improvement (i.e., less than 0.0001) between the costs of the solutions of two successive iterations. Here, we used the Euclidean distance as the measure of distance.

Class I Instances

There are four sets of test data which contain 50, 287, 654, and 1060 fixed points, respectively (see Brimberg *et al.,* 2000). The proposed methods are implemented using the number of open facilities (*M*) to be from 2 to 25 for the 50-fixed points and 5 to 50 with an increment of 5 for the other three data sets. The demand of all data sets is set to unity except the 287 fixed points.

As these data sets do not have a capacity of the facilities, we set the capacity of a facility to be the average demand of all customers, i.e. $b = \left\lceil \sum_{j=1}^{n} w_j \Big/ M \right\rceil$, where $\lceil x \rceil$ is the smallest integer greater than or equal to *x*. This is also the setting used in Zainuddin and Salhi (2007) and Luis *et al.* (2009) whose results will be used for comparison purposes. It is worth noting that there would be cases where the total capacity of the facilities (i.e. *Mb*) would be larger than the total customer demand ($\sum_{j=1}^{n} w_j$). In this situation, (i.e., if $Mb > \sum_{j=1}^{n} w_j$), a dummy customer with a 0 transportation cost and a demand equals to $\left(Mb - \sum_{j=1}^{n} w_j \right)$ will be used. This dummy customer will only contribute to the search at the transportation problem phase and will not be considered at either the location or the allocation phases.

We take the solutions of the MSWP given by Brimberg *et al.* (2000) as 'lower bounds' for the capacitated problem. These solutions are optimal for the 50 and 287 fixed points data sets and the best known solutions for the others. Note that these 'lower bounds' may not be valid for the larger problems where the optimal solutions are unknown but we consider these values as good reference points. Therefore, for each instance the percentage deviation (*Dev*(%)) is calculated based on these 'lower bounds' as follows:

$$Dev(\%) = \frac{F_{best} - F_{LB}}{F_{LB}} \times 100 \qquad (16)$$

where F_{best} is the best solution cost obtained by the proposed algorithms and F_{LB} denotes the 'lower bound' or 'best' cost for the uncapacitated problem. In addition, the overall average (OAV) over all instances for each of the four test data is also recorded.

To our knowledge, the results of the heuristic of Zainuddin and Salhi, ZS for short, and of Luis *et al*. (2009), LSN for short, are the only ones available for direct comparison for these instances. The ATL results are not reported here as this original method was already outperformed by ZS. As there is a large number of instances in this class (55 in total), we only provide the summary results in Table 1. However, for further references and future benchmarking, we also give the detailed results in Appendix A under Tables A1 to A4. The bold numbers in Table 1 indicate the best deviation and the bold numbers in the brackets refer to the number of instances where a best solution is found, including ties. Based on these empirical results, our proposed methods appeared to outperform ZS with respect to both the solution quality and computing times.

Table 1. Comparison of Overall Average Results for Class I Instances

Data Sets	ZS		LSN		GALNP		GALNPL (GALNP with learning)		Presence of Restricted Regions			
									GALNPRR (no learning)		GALNPLRR (with learning)	
	Dev (%)	CPU (secs)	Dev (%)	CPU (secs)	Dev (%)	CPU (secs)	Dev (%)	CPU (secs)	Dev (%)	CPU (secs)	Dev (%)	CPU (secs)
50 fixed Points (24)	10.2 [5]	4.0	7.52 [5]	1.2	7.52 [9]	15.9	7.41 [9]	16.1	6.50 [**17**]	23.7	**6.47** [**17**]	24.2
287 fixed Points (10)	57.0 [4]	6601.8	56.91 [2]	104.7	56.91 [2]	994.4	56.91 [2]	1004.8	**56.89** [**7**]	1055.8	**56.89** [**7**]	1092.9
654 fixed Points (10)	59.2 [4]	4279.3	56.41 [6]	296.3	56.55 [9]	2925.8	56.50 [6]	2951.7	56.41 [**10**]	2956.9	**56.40** [**10**]	3043.3
1060 fixed Points (10)	4.0 [4]	40625.5	3.99 [3]	1058.0	3.95 [6]	6637.6	3.95 [7]	6816.1	3.93 [6]	7097.0	**3.92** [**9**]	7357.2
OAV	26.79	9540	25.06	27	25.08	1962	25.03	2002	24.60	2068	**24.58**	2139
# Best	[17]		[16]		[26]		[24]		[40]		**[43]**	

[*o*] : *o* instances for which a new best solution is found, including ties
(*p*) : total number of instances in that category
OAV: Overall average

For example, in the case of *n* = 50 customers, our methods reduce the overall average deviation by 35% from ZS and by 13% from LSN. The low improvement (in %) in the larger instances looks less convincing. The ZS heuristic was run on a Sun Enterprise workstation dual processor at 300 MHz 450 running Solaris 2.6. This machine is approximately five times slower than ours, according to Dongarra (2008). Taking into account the comparative speed of the machines, our algorithms are found to be relatively much faster than ZS, except for the smallest instances.

It is also observed that for M = 25 in the case of 50 fixed points, the deviation between the 'lower bound' and the total cost is very high approximately by 71%. This case exists because each facility happens to serve only two customers and in some circumstances one of the customers is forced to be served by a farther facility as the nearest facility cannot satisfy its demand. Though the percentage deviation can in the larger instances, be particularly high when based on the 'lower bound', in our view such information, still provides a clear and consistent reference point for comparison purposes.

Class II Instances

There are 18 instances for which the number of customers varies between 2 and 30 and the number of facilities from 2 to 10. Instances 1 – 12 are from Al-Loughani (1997) and instances 15 – 20 are from Sherali *et al.* (2002) who created these from instance 12. These instances differ from Class I, being much smaller, but having different capacities. Sherali *et al.* (2002) found optimal solutions to instances 1 – 8 and 15. For the remaining instances, their results are the best found so far. We use these results as benchmarks. For completeness, we also report the heuristic solutions from Aras *et al.* (2007 a, b) which are found using a PC with a 3.2 GHz Pentium IV processor and 2 GB RAM and a 1.7 GHz Pentium Centrino processor and 256 Mb RAM respectively. Our machine is aproximately twice slower than a 3.2 GHz Pentium IV and has the same speed as the 1.7 GHz Pentium Centrino (Dongarra, 2008).

Given that GALNPRR and GALNPLRR were the best performers when compared to our other variants, for simplicity we record their solutions only. Note that instances 13 and 14 are not considered here as these are also omitted from Aras *et al.* (2007 a, b) being not in that unpublished thesis.

The comparison of all results is given in Table 2. The solutions of our proposed methods are found to be very close to the optimal solutions. The overall average deviation found by GALNPRR and GALNPLRR are 0.04% and 0.03% respectively. In addition, the computing times for our heuristics are almost negligible (less than 0.01 second) and hence compete favourably against other methods that used similar machine speed. In other words, our heuristics could be used, as shown in Class I, to tackle larger instances without any computational handicaps when compared to the others.

Though the *p*-median based approach in Aras *et al.* (2007a) appears to perform well while using a small amount of computing time for small problems, for larger instances this may not be the case as shown by Luis *et al.* (2009). The main reason is that for larger instances, solving the discrete *p*-capacitated facility location problem to optimality is likely to require a significant amount of computational effort.

Table 2. Comparison of the Results for Class II Instances

Instance	(*M*, *n*)	Sherali et al. (2002)		Aras et al. (2007a)		Aras et al. (2007b)#		LSN**	GALNPRR**	GALNPLRR**
		Best Known	CPU (secs)	Dev (%)	CPU (secs)	Dev (%)	CPU (secs)	Dev (%)	Dev (%)	Dev (%)
1	(2, 2)	0.00	0.04	**0.00**	0.02	--	--	**0.00**	**0.00**	**0.00**
2	(2, 4)	247.28	0.20	**0.00**	0.02	**0.00**	1	**0.00**	**0.00**	**0.00**
3	(2, 4)	214.34	0.90	**0.00**	0.03	0.01	1	0.01	0.01	0.01
4	(3, 5)	24.00	2.30	**0.00**	0.03	**0.00**	10	**0.00**	**0.00**	**0.00**
5	(3, 5)	73.96	2.00	**0.00**	0.02	**0.00**	13	0.01	0.01	0.01
6	(3, 9)	221.40	66.40	**0.00**	0.06	**0.00**	95	0.01	0.01	0.01
7	(3, 9)	871.62	42.20	**0.00**	0.05	**0.00**	81	0.01	0.01	0.01
8	(4, 8)	609.23	360.00	**0.00**	0.06	**0.00**	163	0.01	0.01	0.01
9	(5, 15)	8169.79	1380.00	**0.00**	0.33	**0.00**	3823	**0.00**	**0.00**	**0.00**
10	(5, 20)	12846.87	8040.00	**0.00**	0.52	**0.00**	5479	**0.00**	**0.00**	**0.00**
11	(5, 20)	1107.18	4380.00	**0.00**	0.92	**0.00**	8217	0.02	0.01	0.01
12	(5, 30)	23990.04	18960.00	**0.00**	3.86	--	--	**0.00**	**0.00**	**0.00**
15	(5, 10)	2595.47	480.00	**0.00**	0.14	**0.00**	647	0.14	0.11	0.11
16	(6, 10)	7797.21	540.00	**0.00**	0.20	**0.00**	1147	0.13	0.13	0.11
17	(7, 10)	6967.90	18900.00	0.01	0.27	**0.00**	1782	0.11	0.09	0.09
18	(8, 10)	1564.46	28080.00	**0.00**	0.31	**0.00**	3287	0.16	0.11	0.11
19	(9, 10)	3250.68	720.00	**0.00**	0.38	**0.00**	4778	0.10	0.11	0.07
20	(10, 10)	7719.00	27720.00	**0.00**	0.16	0.01	10087	0.19	0.11	0.01
OAV				**0.00**	0.41	**0.00**	2476	0.05	0.04	0.03
# Best				**19**		16		6	6	6

: Instances 1 and 12 are not reported in their paper.
** : CPU time is less than 0.01 seconds and hence this is not reported here.
OAV: Overall average deviations

Class III Instances

These Class III instances are an adaptation of Class I instances. The aim is to evaluate the performance of GALNPRR and GALNPLRR for the larger problems. There are four data sets as in Class I, i.e. the case of n = 50, 287, 654, and 1060 customer points. The customer locations for each data set are the same as those in Class I except here we generate a set of capacities for each data set. For the number of open facilities (M) equal to 5, three set of capacities are generated and for $M > 5$ five set of different capacities are used. For the case of n = 50, we generate randomly capacities in the interval of [1,20], for n = 287, [50, 2100], for n = 654, [5, 164], and for n = 1060 it is [5, 250]. The detailed data can be collected from Luis (2008). In the case of n = 50 customers, the number of open facilities (M) is set to vary from 5 to 15 with an increment of 5, for n = 287, M starts from 5 to

30 with an increment of 5 until *M* = 20 and then an increment of 10. For the other two data sets, *M* is set to be from 10 to 50 with an increment of 10. The results of these data sets are given in Table 3. For comparison purposes, we also present the best results from LSN (i.e., LSN with dynamic radius). It can be observed that GALNPLRR outperforms LSN and GALNPRR for every instance. The **bold** numbers highlight the new best found solutions.

Table 3. Comparison of results for Class III Instances

n	*M*	Lower Bound	LSN		GALNPRR		GALNPLRR	
			Dev (%)	CPU (sec)	Dev (%)	CPU (sec)	Dev (%)	CPU (sec)
50 fixed Points	5	72.24	15.50	2	**15.23**	2	**15.23**	3
	10	41.69	33.18	2	30.27	11	**30.23**	13
	15	27.63	43.22	3	**38.77**	24	**38.77**	29
	OAV		30.63	2.33	28.09	12.33	**28.08**	15.00
287 fixed Points	5	9715.63	17.42	18	**16.10**	20	**16.10**	28
	10	6705.04	31.19	30	**28.48**	49	**28.48**	57
	15	5224.70	30.28	40	24.95	120	**23.15**	142
	20	4148.84	39.25	65	**32.97**	275	**32.97**	319
	30	2716.91	59.66	123	53.18	775	**50.27**	835
	OAV		35.56	55.20	31.14	247.80	**30.19**	276.20
654 fixed Points	10	115339.03	58.02	58	45.52	106	**45.51**	131
	20	63389.02	49.86	156	**42.15**	717	**42.15**	787
	30	44705.19	60.97	326	51.12	2214	**50.55**	2499
	40	35704.41	67.56	576	61.96	4327	**56.88**	4651
	50	29338.01	71.43	986	**67.35**	9688	**67.35**	11001
	OAV		61.57	420.40	53.62	3410.40	**52.49**	3813.80
1060 fixed Points	10	1249564.75	14.06	187	**12.66**	252	**12.66**	381
	20	828802.00	19.06	550	**17.67**	1692	**17.67**	1825
	30	638263.00	23.27	1176	20.11	5375	**20.08**	5741
	40	529866.19	35.21	2012	32.53	11384	**32.13**	11884
	50	453164.00	51.87	3192	48.44	21514	**48.21**	22191
	OAV		28.69	1423.40	26.28	8043.40	**26.15**	8404.40
OAV			37.70	528	35.53	3252	**29.10**	3473
# Best			0		9		**18**	

OAV: Overall average deviations over all instances and over each of the 4 data set

It is worth noticing that all the best results are found by GALNPLRR. In addition, this method produces better results on average with a 5 to 13% improvement when compared with the one given by LSN. However, it is worth pointing out that LSN appears to be relatively faster than our GRASP-based methods requiring about 15 to 20% of their total CPU time only.

7. Conclusions and Future Research

In this study, a guided reactive GRASP to tackle the capacitated multi-source Weber problem is proposed. The construction of RCL is guided by using a functional threshold parameter α namely a linear function. A learning process is also embedded into the search to define the bounds of the parameter α. To guide the search further, we introduced restricted regions to avoid generating locations that are sited too close to each other by having dynamically adjusted radiuses of these regions. This adjustment is carried out to cater for the capacity of the facility as well as the demand of the allocated customers. Three classes of data sets are used, those with large instances but having facilities with constant capacities, the small ones that allow the facilities to have different capacities from which the optimal solutions exist and the new large ones designed by the authors to allow for different capacities. The results from all these instances show that our proposed methods produce competitive results when compared against the optimal solutions and recent heuristics.

The following research directions may be worthy of investigation in the future. The restricted region could include other shapes such as rectangular or square. In this study, the runs within GRASP are used independently, but the information from one run to the next could be taken advantage of and used efficiently to guide the search better in subsequent runs. This concept of memory is useful not only for this type of location problems but also in a variety of combinatorial problems. From a practical point of view, the impact of introducing the opening cost of a facility (fixed cost) is important and could be explored. The fixed cost can be constant, dependent on the location (zone-dependent as in Brimberg and Salhi (2005)), or throughput-related. Another direct modification to the existing problem would be the case of restricting a customer to be served by one facility only as in the case of the single source type problem.

Acknowledgments- The authors would like to thank both referees for their constructive comments which improved both the content as well as the presentation of the paper.

References

1. Al-Loughani, I., 1997. Algorithmic Approaches for Solving the Euclidian Distance Location-Allocation Problems. *Ph.D. Dissertation*, Industrial and System Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
2. Aras, N., Ozkisacik, K. C. and Altinel, I. K., 2006. Solving the Uncapacitated Multi-Facility Weber Problem by Vector Quantization and Self-organizing Maps. *Journal of the Operational Research Society* 57, 82-93.
3. Aras, N., Altınel, I. K. and Orbay, M., 2007a. New Heuristic Methods for the Capacitated Multi-Facility Weber Problem. *Naval Research Logistics* 54, 21-32.
4. Aras, N., Yumusak, S. and Altınel, I. K., 2007b. Solving the Capacitated Multi-Facility Weber Problem by Simulated Annealing, Threshold Accepting and Genetic Algorithms. In Doerner, K. F., Gendreau, M., Greistorfer, P., Gutjahr, W. J., Hartl, R. F. and Reimann, M., editors. *Metaheuristics: Progress in Complex Systems Optimization*, Springer, 91-112.
5. Aras, N., Orbay, M. and Altinel, I. K., 2008. Efficient Heuristics for the Rectilinear Distance Capacitated Multi-Facility Weber Problem. *Journal of the Operational Research Society* 59, 64-79.
6. Bazaraa, M., Jarvis, J. J. and Sherali, H. D., 2005. *Linear Programming and Network Flows*, third edition, WileyBlackwell.
7. Bischoff, M. and Klamroth, K., 2007. An Efficient Solution Method for Weber Problem with Barriers Based on Genetic Algorithms. *European Journal of Operational Research* 177, 22-41.
8. Brimberg, J. and Love, R.F., 1993. Global convergence of a generalized iterative procedure for the minimum location problem with l_p distances. *Operations Research* 41: 1153-1163.
9. Brimberg, J., Hansen, P., Mladenović, N. and Taillard, E. D., 2000. Improvements and Comparison of Heuristics for Solving the Uncapacitated Multisource Weber Problem. *Operations Research* 48, 444-460.
10. Brimberg, J. and Salhi, S., 2005. A Continuous Location-Allocation Problem with Zone-Dependent Fixed Cost. *Annals of Operations Research* 136, 99-115.
11. Brimberg, J., Hansen, P., Mladenović, N. and Salhi, S., 2008. A Survey of Solution Methods for the Continuous Location-Allocation Problem. *International Journal of Operations Research* 5, 1-12.
12. Canbolat, M. S. and Wesolowsky, G. O., 2010. The Rectilinear Distance Weber Problem in the Presence of a Probabilistic Line Barrier. *European Journal of Operational Research* 202, 114-121.
13. Cooper, L., 1964. Heuristic Methods for Location-Allocation Problems. *SIAM Review* 6, 37-53.
14. Cooper, L., 1972. The Transportation-Location Problem. *Operations Research* 20, 94-108.
15. Cooper, L., 1975. The Fixed Charge Problem – I: A New Heuristic Method. *Computers and Mathematics with Applications* 1, 89-95.
16. Cooper, L. 1976. An Efficient Heuristic Algorithm for the Transportation-Location Problem. *Journal of Regional Science* 16, 309-3015.
17. Daskin, M. S., 2008. What you should know about location modelling. *Naval Research Logistics* 55, 283-294.
18. Delmaire, J., Diaz, A., Fernandez, E. and Ortega, M., 1999. Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR* 37, 194-225.
19. Deng, Y. and Bard, J. F., 2010. GRASP with path relinking for capacitated clustering. *Journal of Heuristics* (forthcoming), doi: 10.1007/s10732-010-9129-z

20. Dongarra, J. J., 2008. Performance of Various Computers using Standard Linear Equations Software. *Technical Report* available on line at http://www.netlib.org/benchmark/performance.ps.
21. Feo, T. A. and Resende, M. G. C., 1989. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Operations Research Letters*, 8, 67-71.
22. Feo, T. A. and Resende, M. G. C., 1995. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6, 109-133.
23. Gamal, M. D. H. and Salhi, S., 2001. Constructive Heuristics for the Uncapacitated Location–Allocation Problem. *Journal of the Operational Research Society* 51, 1233-1240.
24. Gamal, M. D. H. and Salhi, S., 2003. A Cellular Heuristic for the Multisource Weber Problem. *Computers and Operations Research* 30, 1609-1624.
25. Gong, D., Gen, M., Yamazaki, G. and Xu, W., 1997. Hybrid Evolutionary Method for Capacitated Location-Allocation Problem. *Computers and Industrial Engineering* 33, 577-580.
26. Hansen, P., Mladenović, N. and Taillard, E. D., 1998. Heuristic of the Multisource Weber Problem as a *p*-Median Problem. *Operation Research Letters* 22, 55-62.
27. Hart, J. P. and Shogan, A. W., 1987. Semi-greedy Heuristics: An Empirical Study. *Operations Research Letters*, 6, 107-114.
28. Katz, I. N. and Cooper, L., 1981. Facility Location in the Presence of Forbidden Regions, I: Formulation and the Case of Euclidean Distance with One Forbidden Circle. *European Journal of Operational Research* 6, 166-173.
29. Kuehn, A. A. and Hamburger, M. J., 1963. A Heuristic Program for Locating Warehouse. *Management Science* 9, 643-666.
30. Kuhn, H.M. , 1973. A note on Fermat’s problem. *Mathematical Programming* 3, 193-209.
31. Luis, M., 2008. Meta-heuristics for the Capacitated Multi-Source Weber Problem, *Ph.D. Thesis*, University of Kent, UK.
32. Luis, M., Salhi, S. and Nagy, G., 2009. Region-Rejection Based Heuristics for the Capacitated Multi-Source Weber Problem, *Computers and Operations Research* 36, 2007-2017.
33. Megiddo, N. and Supowit, K. J., 1984. On the Complexity of Some Common Geometric Location Problems, *SIAM J. Comp.* 13, 182-196.
34. Mirchandani, P. B. and Francis, R. L., 1990. *Discrete Location Theory*, New York: Wiley.
35. Pitsoulis, L. S. and Resende, M. G. C., 2002. Greedy Randomized Adaptive Search Procedures. In Pardalos, P. M. and Resende, M. G. C., editors. *Handbook of Applied Optimization*. Oxford University Press, 168-181.
36. Resende, M. G. C. and Ribeiro, C. G., 2003. Greedy Randomized Adaptive Search Procedures. In Glover, F. and Kochenberger, G. A., editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 219-249.
37. Resende, M. G. C. and Ribeiro, C. G., 2005. GRASP with path-relinking: recent advances and applications. In Ibaraki T., Nonobe, K. and Yagiura, M., editors. *Metaheuristics: progress as real problem solvers*, Springer, NY, 29-63.
38. Rios-Mercado, R. Z. and Fernandez, E., 2009. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers and Operations Research* 36, 755-776.
39. Rosing, K. E., 1992. The Optimal Location of Steam Generators in Large Heavy Oil Fields. *American Journal of Mathematical and Management Sciences* 12, 19-42.
40. Salhi, S. and Gamal, M. D. H., 2003. A GA Based Heuristic for the Uncapacitated Continuous Location-Allocation Problem. *Annals of Operations Research* 123, 203-222.
41. Sherali, H. D., Shetty, C. M., 1977. The Rectilinear Distance Location-Allocation Problem. *AIIE Transaction* 9, 136-143.

42. Sherali, H. D. and Tuncbilek, C. H., 1992. A Squared-Euclidean Distance Location-Allocation Problem. *Naval Research Logistics* 39, 447-469.
43. Sherali, H. D., Ramachandran, S. and Kim, S., 1994. A Localization and Reformulation Discrete Programming Approach for the Rectilinear Distance Location-Allocation Problem. *Discrete Applied Mathematics* 49, 357-378.
44. Sherali, H. D., Al-Loughani, I. and Subramanian, S., 2002. Global Optimization Procedures for the Capacitated Euclidean and ℓ_p Distance Multifacility Location-Allocation Problem. *Operations Research* 50, 433-448.
45. Taillard, E. D., 2003. Heuristics Methods for Large Centroid Clustering Problems. *Journal of Heuristics* 9, 51-73.
46. Zainuddin, Z. M. and Salhi, S., 2007. A Perturbation-Based Heuristic for the Capacitated Multisource Weber Problem. *European Journal of Operational Research* 179, 1194-1207.

Appendix A. Detailed Deviation (%) for Class I Instances

Table A1. Comparison for All the Results for the 50 Customers Problem

M	Lower Bound	ZS		LSN		GFAP		GUAP		GALNP		GALNPL		GALNPRR		GALNPLRR	
		Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)
2	135.52	**0.83**	0.56	0.84	1	0.84	0	0.84	1	0.84	0	0.84	1	0.84	1	0.84	1
3	105.21	**1.02**	0.70	1.06	0	1.06	1	1.06	0	1.06	1	1.06	1	1.06	1	1.06	1
4	84.15	2.76	0.67	**2.75**	1	**2.75**	1	**2.75**	1	**2.75**	0	**2.75**	1	**2.75**	1	**2.75**	2
5	72.24	5.94	1.14	**5.93**	0	**5.93**	1	**5.93**	1	**5.93**	2	**5.93**	2	**5.93**	3	**5.93**	2
6	60.97	**0.85**	0.76	0.88	1	0.88	1	0.88	2	0.88	1	0.88	1	0.88	4	0.88	3
7	54.50	**3.35**	2.17	3.37	1	3.37	2	3.37	2	3.37	2	3.37	2	3.37	4	3.37	3
8	49.94	2.76	1.06	**2.40**	0	**2.40**	3	**2.40**	2	**2.40**	3	**2.40**	3	**2.40**	5	**2.40**	5
9	45.69	4.05	3.05	**3.46**	1	**3.46**	3	**3.46**	4	**3.46**	3	**3.46**	3	**3.46**	6	**3.46**	8
10	41.69	15.72	4.21	15.67	1	**15.54**	5	**15.54**	4	**15.54**	5	**15.54**	5	**15.54**	7	**15.54**	10
11	38.02	6.57	3.42	**6.07**	1	**6.07**	6	6.08	6	**6.07**	6	**6.07**	6	**6.07**	9	**6.07**	10
12	35.06	1.98	1.40	2.00	1	**1.99**	7	2.55	8	**1.99**	7	**1.99**	8	**1.99**	12	**1.99**	12
13	32.31	8.83	2.81	8.80	2	8.80	10	**8.79**	10	8.80	10	8.80	10	**8.79**	14	8.80	13
14	29.66	6.62	2.86	4.69	1	5.47	12	**4.68**	11	4.69	12	4.69	12	**4.68**	16	**4.68**	15
15	27.63	6.49	3.68	2.60	2	2.80	14	2.60	14	**1.53**	14	**1.53**	14	**1.53**	20	**1.53**	20
16	25.74	5.93	4.41	2.15	1	2.60	17	2.13	17	1.01	17	1.01	17	**0.46**	25	**0.46**	26
17	23.99	7.92	6.47	7.60	1	**2.38**	20	7.59	20	7.59	20	7.59	20	7.59	30	7.59	28
18	22.29	6.89	6.85	5.59	2	7.59	22	5.59	22	**5.58**	22	**5.58**	22	**5.58**	32	**5.58**	34
19	20.64	7.32	5.97	5.44	1	5.58	24	**4.85**	24	5.03	24	5.03	25	5.03	35	5.03	37
20	19.36	17.43	9.56	4.47	1	6.24	27	4.70	27	6.06	27	5.22	27	**3.16**	40	**3.16**	40
21	18.08	16.41	9.53	7.31	2	3.16	31	3.01	31	2.44	31	2.44	31	**1.36**	46	**1.36**	48
22	16.82	13.60	6.21	5.84	1	3.31	36	5.89	36	3.51	36	3.51	36	**1.20**	55	**1.20**	55
23	15.61	16.12	6.35	4.08	2	5.11	40	3.91	40	4.90	40	3.83	40	**0.49**	60	**0.49**	61
24	14.44	15.35	5.92	4.46	2	7.35	46	9.91	46	5.96	46	5.96	46	**0.26**	68	**0.26**	70
25	13.30	**70.96**	6.25	73.05	2	74.79	52	76.53	51	79.00	52	78.31	53	71.61	76	**70.96**	77
OAV		10.24	4.00	7.52	1.17	7.76	15.88	7.71	15.83	7.52	15.88	7.41	16.08	6.50	23.75	**6.47**	24.21

Table A2. Comparison for All the Results for the 287 Customers Problem

M	Lower Bound	ZS		LSN		GFAP		GUAP		GALNP		GALNPL		GALNPRR		GALNPLRR	
		Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)
5	9715.63	**7.90**	56.57	7.92	11	7.92	11	7.92	12	7.92	12	7.92	12	7.92	12	7.92	14
10	6705.04	**25.77**	539.11	25.80	17	25.80	34	25.80	34	25.80	33	25.80	35	25.80	37	25.80	41
15	5224.70	**33.82**	1837.17	33.89	25	33.99	93	33.89	93	33.89	92	33.99	94	33.89	111	33.99	118
20	4148.84	38.99	2532.21	**38.48**	38	**38.48**	224	**38.48**	225	**38.48**	223	**38.48**	231	**38.48**	242	**38.48**	251
25	3348.71	44.37	3442.54	44.33	64	44.32	427	44.33	427	44.39	424	44.33	429	**44.31**	462	**44.31**	478
30	2716.91	54.23	6265.81	54.15	85	54.15	724	54.15	722	54.15	720	54.15	730	**54.14**	770	**54.14**	792
35	2238.18	69.27	8310.36	69.24	125	**69.23**	1140	69.33	1129	**69.23**	1135	**69.23**	1151	**69.23**	1190	**69.23**	1202
40	1900.84	84.96	10948.99	**84.33**	151	84.34	1679	84.34	1665	84.41	1671	84.35	1688	**84.33**	1680	**84.33**	1688
45	1630.31	**94.82**	12169.82	94.92	219	94.92	2382	94.92	2362	94.92	2368	94.92	2389	**94.82**	2454	**94.82**	2530
50	1402.58	115.94	19814.90	116.06	312	115.95	3281	115.95	3255	115.95	3266	115.95	3289	**115.93**	3600	**115.93**	3815
OAV		57.01	6601.76	56.91	104.70	56.91	995.50	56.91	992.40	56.91	994.40	56.91	1004.80	**56.89**	1055.80	**56.89**	1092.90

Table A3. Comparison for All the Results for the 654 Customers Problem

M	Lower Bound	ZS		LSN		GFAP		GUAP		GALNP		GALNPL		GALNPRR		GALNPLRR	
		Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)
5	209068.80	**54.00**	62.67	**54.00**	18	**54.00**	20	**54.00**	20	**54.00**	20	54.00	21	**54.00**	21	**54.00**	25
10	115339.03	**42.81**	136.94	**42.81**	31	**42.81**	70	**42.81**	71	**42.81**	72	42.81	77	**42.81**	75	**42.81**	81
15	80177.04	**67.69**	852.92	**67.69**	64	**67.69**	272	**67.69**	270	**67.69**	275	67.69	327	**67.69**	280	**67.69**	355
20	63389.02	69.37	1086.59	**69.36**	107	**69.36**	662	**69.36**	664	**69.36**	663	**69.36**	677	**69.36**	680	**69.36**	726
25	52209.51	47.52	3842.79	47.52	180	47.52	1266	47.52	1265	**47.51**	1264	**47.51**	1277	**47.51**	1291	**47.51**	1331
30	44705.19	86.77	852.18	**76.33**	247	**76.33**	2131	**76.33**	2126	**76.33**	2132	**76.33**	2155	**76.33**	2160	**76.33**	2199
35	39257.27	**78.13**	5772.53	**78.13**	354	78.22	3340	**78.13**	3324	**78.13**	3334	**78.13**	3357	**78.13**	3370	**78.13**	3455
40	35704.41	44.25	4740.56	43.85	517	43.88	4956	**43.84**	4932	**43.84**	4949	**43.84**	5012	**43.84**	5026	**43.84**	5112
45	32306.97	59.23	5112.34	55.28	638	55.29	7000	55.28	6968	**55.26**	7002	**55.26**	7021	**55.26**	7045	**55.26**	7265
50	29338.01	41.96	20333.38	29.13	807	30.68	9544	30.68	9488	30.61	9547	30.11	9593	**29.13**	9621	**29.13**	9884
OAV		59.17	4279.29	56.41	296.30	56.58	2926.10	56.56	2912.80	56.55	2925.80	56.50	2951.70	56.41	2956.90	**56.40**	3043.30

Table A4. Comparison for All the Results for the 1060 Customers Problem

M	Lower Bound	ZS		LSN		GFAP		GUAP		GALNP		GALNPL		GALNPRR		GALNPLRR	
		Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)	Dev. (%)	CPU (sec)
5	1851879.88	**1.06**	370.85	**1.06**	60	**1.06**	58	**1.06**	56	**1.06**	60	**1.06**	66	**1.06**	58	**1.06**	72
10	1249564.75	**3.11**	1167.35	**3.11**	101	**3.11**	199	**3.11**	198	**3.11**	201	**3.11**	231	**3.11**	201	**3.11**	264
15	980132.13	**1.63**	4063.94	**1.63**	255	**1.63**	722	**1.63**	716	**1.63**	731	**1.63**	791	**1.63**	742	**1.63**	845
20	828802.00	3.42	5116.90	3.35	436	3.35	1632	3.35	1616	3.35	1646	3.35	1751	3.35	1659	3.35	1833
25	722061.19	3.87	30551.64	3.91	662	3.87	3036	3.87	2985	3.86	3048	**3.85**	3056	3.86	3063	**3.85**	3261
30	638263.00	**3.92**	45191.83	3.95	1002	3.95	4998	3.93	4955	**3.92**	4979	**3.92**	5111	3.93	5034	**3.92**	5311
35	577526.63	3.35	21713.60	3.36	1365	3.32	7740	3.32	7680	**3.31**	7734	**3.31**	7900	**3.31**	7839	**3.31**	8253
40	529866.19	6.02	47253.61	6.16	1700	6.05	11124	6.05	11974	6.05	11154	6.10	12354	**5.99**	11242	**5.99**	12754
45	489650.00	7.83	120072.31	7.88	2204	**7.74**	15617	**7.74**	15159	**7.74**	15605	**7.74**	15513	7.76	15722	**7.74**	17125
50	453164.00	5.87	130753.06	5.50	2797	5.52	21182	5.50	20912	5.50	21218	5.45	21388	**5.29**	25410	**5.29**	23854
OAV		4.01	40625.51	3.99	1058.00	3.96	6630.80	3.96	6625.10	3.95	6637.60	3.95	6816.10	3.93	7097.00	**3.92**	7357.20