An Adaptive Perturbation-Based Heuristic: An Application to the Continuous p-Centre Problem*

Abdalla Elshaikh[a,d], Said Salhi[a], Jack Brimberg[b], Nenad Mladenović[c], Becky Callaghan[a] and Gábor Nagy[a]

[a] Centre for Logistics and Heuristic Optimisation (CLHO), Kent Business School, University of Kent, Canterbury, UK
{ae201, s.salhi, bc349, g.nagy}@kent.ac.uk
[b] Department of Mathematics and Computer Science, Royal Military College of Canada, Kingston, ON K7K 7B4, Canada
Jack.Brimberg@rmc.ca
[c]LAMIH, Universite de Valenciennes, France
nenad.mladenovic@univ-valenciennes.fr
[d]Faculty of Economics, University of Misurata, Misurata, Libya.

Abstract

A self-adaptive heuristic that incorporates a variable level of perturbation, a novel local search and a learning mechanism is proposed to solve the p-centre problem in the continuous space. Empirical results, using several large TSP-Lib data sets, some with over 1300 customers with various values of p, show that our proposed heuristic is both effective and efficient. This perturbation metaheuristic compares favourably against the optimal method on small size instances. For larger instances the algorithm outperforms both a multi-start heuristic and a discrete-based optimal approach while performing well against a recent powerful VNS approach. This is a self-adaptive method that can easily be adopted to tackle other combinatorial/global optimisation problems. For benchmarking purposes, the medium size instances with 575 nodes are solved optimally for the first time, though requiring a large amount of computational time. As a by-product of this research, we also report for the first time the optimal solution of the vertex p-centre problem for these TSP-Lib data sets.

Keywords- p-centre problem, continuous space, perturbation search, adaptive search, large instances, optimal solutions.

* This research has been supported in part by the UK Research Council EPSRC (EP/I009299/1), the Natural Sciences & Engineering Research Council of Canada Discovery Grant (NSERC #20541 – 2008), the Russian Federation grant RFS 14-41-00039, the National Council for Scientific and Technological Development - CNPq/Brazil grant number 400350/2014-9, and the Spanish Ministry of Economy and Competitiveness, research project MTM2015-70260-P.

1 Introduction

Continuous location problems are concerned with the location of one or more facilities in the plane. These are characterised by the number of possible sites being infinite and hence the unconstrained location of new facilities can be anywhere. In other words, any point is considered as a potential location for a new facility. The objective of the p-centre problem is to minimise the maximum distance between all customers (demand points or fixed points) and their nearest facilities. This problem is particularly useful in locating emergency facilities, such as fire stations, police stations and hospitals, where it is aimed to minimise the longest response time.

For completeness, we cite a few $p-$centre related real life applications spanning over the last 25 years. One of the earliest applications considers the location of fifteen fire stations in the Belgian rural province of Luxembourg. This problem was investigated by Richard, Beguin and Peeter's [24] who used villages, sparsely populated hamlets and some roads in the country side as demand points, some of which also served as potential sites. The location of a number of health resources such as geriatric and diabetic health care clinics in the rural area of Burgos in Spain was examined by Pacheco & Casado [22] using scatter search. A study to locate a number of bicycle stations in the city of Isfahan, Iran, was conducted by Kavesh and Nasr [18] using harmony search. A real life application that aims to minimise the number of emergency warning sirens in Dublin (Ohio) was explored by Wei et al. [30] who adopted an enhanced Voronoi-based approach to cover the entire area with the minimum number of facilities. A humanitarian aid problem to locate a number of urgent relief distribution centres to help with the casualties due to an earthquake in Taiwan that measured at 7.3 on the Richter scale, and caused over 2500 deaths and 8000 injuries, was recently investigated by Lu [19] using simulated annealing.

The continuous (or planar) p-centre problem has a succinct geometrical interpretation. For example, the single unweighted facility location problem (i.e., $p=1$) corresponds to finding the smallest circle that encloses all n points (customers), with the centre being the location of the new facility. Equivalently, the continuous p-centre problem ($p>1$) aims to cover a set of customers in the plane with p circles where the radius of the largest circle is minimised.

The (weighted) p-centre problem can be formulated as follows (Drezner [6]).

$$\underset{X_1,...,X_p}{\text{Minimize}}\left\{\underset{1\le i\le n}{\text{Max}}\,[\underset{1\le j\le p}{\text{Min}}\, D_i(X_j)]\right\}$$

where

$X_j = (x_j, y_j) \in \Re^2$: the coordinates of facility $j\ (j = 1,..., p)$

(a_i, b_i): the coordinates of demand point $i\ (i = 1,..., n)$

w_i: the weight associated with demand point $i\ (i = 1,..., n)$

$D_i(X_j) = w_i[(x_j - a_i)^2 + (y_j - b_i)^2]^{1/2}$: the weighted Euclidean distance between the j^{th} facility and the i^{th} demand point $(i = 1,..., n;\ j = 1,..., p)$.

For variable p, the continuous p-centre problem is known to be NP-hard (see Megiddo and Supowit [20]), whereas for fixed p, Drezner [6] shows that the problem can be solved in $O(n^{2p+4})$ though it is computationally unattractive for large p.

The single facility minimax location problem (1-centre) in the continuous space has a long history, having been posed originally in 1857 by the English mathematician James Joseph Sylvester (1814-1897) who also proposed in 1860 an algorithm to solve it. Elzinga and Hearn [12] proposed an efficient geometrical-based algorithm for solving optimally the problem. Other authors attempted some enhancements to speed up the search, such as Xu et al. [31] and Elshaikh et al. [11] and references therein. For more details on the continuous 1-center problem including a fascinating history on this topic, the reader will find the chapter by Drezner [9] to be informative.

Drezner [7] proposed two algorithms for the solution of the two-centre and two-median location problems with Euclidean distances on the plane. The idea is that the two customer sets in any solution can be separated by a straight line (i.e., $\frac{n(n-1)}{2}$ possibilities). Since the optimal facility location in each of the two sets ($p = 1$) can be easily found due to the convexity of the objective function, the problem reduces to finding an efficient way of defining all these straight lines and hence these corresponding subset pairs.

There is, however, a relatively small number of authors who have studied the p- centre problem; see Plastria [23] and the references therein. One of the commonly used approaches is based on Cooper's [5] locate-allocate procedure. In brief, the idea is to choose initially p facility points randomly or using a heuristic and assign each demand point to its nearest facility

making p subsets. In each cluster the optimal single facility location is found using Elzinga-Hearn or an equivalent method. The allocation is then performed again followed by the optimal solution of p 1-centre problems. This is repeated until there is no improvement in the allocation. Drezner [6] presents two methods, namely, a multi-start similar to Cooper's locate allocate adapted to the p-center problem (referred to as (H1)) followed by a composite heuristic made up of H1 and a post optimiser that allocates the critical points between the clusters (called (H2)). Eiselt and Charlesworth [13] propose three constructive and improvement-based heuristics. Their first one resembles the locate-allocate procedure of Cooper, the second uses the vertex substitution of Teitz and Bart [29] with the critical points used for reallocation, and their third one is based on the drop method. As the latter will be used in our computational results section, we briefly describe it here. The idea is to start with all n demand sites as potential sites and then combine the two nearest points to make up a new centre leading to $n-1$ clusters. This process of exploring the two nearest centers to make up a combined center continues until p clusters with their corresponding centers are found. The 'locate-allocate' process is then activated as an optional improvement step. A more flexible version is to allow a certain number of pairs with their corresponding customers to be explored and the pair corresponding to the combined cluster with the lowest radius is chosen instead of selecting the pair with the closest distance. A control parameter β $(0<\beta\leq 1)$ is introduced to select these pairs with a value of 0.5 empirically shown to produce the best results. This flexible variant, known as STEPDOWN, outperforms their other two methods. Very recently, Elshaikh et al. [11] devise an enhanced version of the Elzinga and Hearn algorithm for the 1-centre problem which is then embedded within a powerful VNS-based heuristic to solve the p-centre problem. The results from H1, H2 and STEPDOWN heuristics will be used alongside those given in [11] for comparison purposes in subsection 5.2.

For the case of area coverage, which can be of interest, for example, to agriculture, environment and mobile phone coverage technology, a Voronoi diagram-based heuristic, using an iterative procedure based on the locate-allocate principle, was proposed by Suzuki and Okabe [28]. This was then applied by Drezner and Suzuki [8] who added a post-optimiser using nonlinear programming to cover a square with p circles. Wei et al. [30] extended the above Voronoi-based approach to account for irregular and non-convex shapes, including the possibility of forbidden regions where the new facilities cannot be sited. Though the area and the point coverage problems are related, these preceding approaches

should not be used directly for point coverage given that the results can be misleading as demonstrated by Murray and Wei [21].

Few papers deal with exact methods for the planar p-center problem. Drezner [6] put forward an interesting idea of enumerating all the maximum sets given a threshold (the radius of the largest circle at a given iteration) to be used within a covering-based model. If the problem is feasible, the obtained feasible solution is then used to get a new threshold. The process is repeated until the covering problem has no feasible solution leading to the current threshold being the optimal solution. Results for small instances up to $n = 40$ and $p = 5$ were tested starting with the initial solution (threshold) found by the Drezner's heuristic H2 [6]. This optimal method will be revisited in the computational results section as it is found to be not as slow as originally mentioned in the literature (see subsection 5.2). Excellent results for both the discrete and the continuous cases are found by Chen and Chen [3] who extended the work of Chen and Handler [4] in several interesting ways. The authors used three types of relaxation methods. One is to solve optimally for a small subset of the original problem, while gradually adding additional demand points (usually the farthest from the service points of the current feasible solution) until the solution becomes feasible for the original problem, and hence, may be considered as the optimal solution of the original problem. Two further relaxations were developed. These include a reverse relaxation where a lower bound is first found which is then gradually increased until the optimal solution is reached, and a binary relaxation where both upper and lower bounds are updated accordingly. The only optimal solutions for the planar p-centre problem reported by the authors are for the TSP data set with $n = 439$. For comparison purposes, these optimal results will also be used in our computational results section (see subsection 5.2).

It is worth noting that the proposed perturbation heuristic is, to our knowledge, the second only metaheuristic that is developed to investigate this class of location problem. This is an adaptive method that can easily be modified to tackle a variety of combinatorial and/or global optimisation problems. In addition, this approach solves large data sets with more than 1300 demand points with encouraging results. For benchmarking purposes, we have also implemented Drezner's optimal method and report, for the first time, the optimal solutions for medium size instances (i.e., $n = 575$) though the computational time required was excessively large especially for small values of p (mostly exceeding 10 hours of CPU time with a few that required nearly 24 hours.

Though the $p-$centre problem can be seen as an old and well-established combinatorial problem, in our view it serves as an interesting and useful base to test innovative ideas which can then be extended and adapted for other related and more complex continuous location problems such as those with restricted non convex regions with and without capacity restriction, presence of fixed cost, just to cite a few.

The contributions of the study include

(i) The design of a powerful perturbation-based metaheuristic that uses an adaptive degree of perturbation and can be adapted to a variety of other combinatorial and global optimisation problems.

(ii) A novel local search that is based on the concept of a 'covering circle' whose neighbourhood is dynamically adjusted.

(iii) The incorporation of learning within the search, which we consider to be an invaluable ingredient in heuristic search design in general and in this new perturbation metaheuristic in particular.

(iv) The generation of high quality results for large planar p-centre problems (some instances with more than 1300 customers) including the optimal solutions for the first time for $n=575$, as well as all the optimal solutions for their discrete counterpart problems.

The paper is organised as follows: The next section discusses the basic perturbation heuristic. In section 3, the two local searches including a novel swap-based scheme using the concept of covering circles are first described, followed by the two new perturbation-based heuristics that use a dynamic level of perturbation. In section 4, learning is introduced within the search. Computational experiments are given in section 5 and our conclusions and suggestions for future research are summarised in the last section.

2 A brief on the basic perturbation-based heuristic

This approach guides the search by introducing some perturbations into the problem. For the p-centre problem these can be achieved by allowing the number of facilities of a solution to go over and under the required number of facilities (p) by a certain value (q). In other words, the solution is allowed to be infeasible in terms of the number of open facilities. In brief, the method works as follows: An initial solution of the p-centre

problem is first found, and then the number of open facilities is allowed to increase to $\bar{p}(\bar{p} = p + q)$ by adding q facilities to the current solution. The removal of q facilities is then performed to reach a solution with p facilities where an intensification of the search is activated. The removal of facilities continues until the problem with $\bar{\bar{p}}$ $(\bar{\bar{p}} = p - q)$ facilities is reached. At this stage the addition of q facilities is performed to get a feasible solution with p open facilities where intensification is activated again. We refer to this shifting as one cycle of the perturbation procedure which is then repeated several times until the maximum computation time allowed (CPU_{Max}) is reached, or the maximum number of cycles without successive improvement is met, whichever comes first.

The reasoning behind this method is that the continual shifting between feasible and infeasible regions acts as a filtering process where the best facilities have the tendency to remain in the promising set. Salhi [25] proposed this metaheuristic for a class of discrete uncapacitated location problems with good results. Hanafi and Freville [15] also adapted a similar approach for solving a class of knapsack problems, while Zainuddin and Salhi [32] modified this methodology to solve the capacitated multisource Weber problem. It can also be noted that the idea of perturbation shares a few similarities with large neighbourhood search proposed by Shaw [27], where the `ruin and build' scheme corresponds to the `drop and add' counterpart, especially when the search goes from p to $p - q$, and then back up to p.

3. The new perturbation-based heuristic

In this study we extend the perturbation metaheuristic given in [25] by

(i) introducing flexibility in the level of perturbation using a variable value of q that is adaptively determined instead of being fixed throughout the search as initially used in the literature [25,32].

(ii) Tailoring the swap, add and drop moves to the p- centre problem.

(iii) Examining two new local searches. One relates to the case when the solution is infeasible (i.e., the number of facilities in the solution is $p \pm s, s = 1,...,q$) where the 'locate-allocate' type procedure, which we refer to as the local search of type1 "LS1", is applied. The second one is used when the solution is feasible (i.e., the number of facilities is p). In this case, a combined local search "LS2" made up of LS1 and a swap-based neighbourhood is adopted.

In (i), we propose two types of perturbation based on flexibility of the level of perturbation where the value of q can be relaxed and made dynamic starting from $q=1$ and increasing to q_{Max}. The first one which we call a gradual perturbation "GRADPERT" aims to add one facility at a time and apply LS1 at $p \pm s, s=1,..., q_{Max}$ whereas the second, which we refer to as the strong perturbation "STRONGPERT", adds all the q facilities in one step, followed by LS1. In addition, GRADPERT uses the first covering circle (CC_1) as its destination cluster when using the add move whereas STRONGPERT adopts a covering circle with a dynamically changing size $(CC_k)_{k=1,...,q_{Max}}$. The definition of CC_k will be given next. In both perturbations LS2 is used whenever the number of facilities is p (i.e., the solution is feasible).

The two local searches (LS1 and LS2) followed by a brief description of the three types of moves (drop, add, swap) are presented in the next subsection, while the GRADPERT and STRONGPERT heuristics will be given in subsections 3.2 and 3.3, respectively.

3.1 The two local searches

a) Local search LS1

LS1 is activated when the number of open facilities is $\bar{p} \in \{p-q,...., p-1, p+1,..., p+q\}$. This procedure is similar to Cooper's `locate allocate procedure' , and is briefly described in the following three mini steps:

(i) Given the $\bar{p}$ facility locations (or centres) $C_j; j=1,..., \bar{p}$, allocate each customer to its nearest centre (breaking ties arbitrarily), and define for each centre j, the subset V_j, as

$$V_j = \{i \in \{1,...,n\} : d(C_j, i) = Min(d(C_k, i), k=1,..., \bar{p})\}$$

(ii) In each subset V_j, determine the optimal centre, C_j, $j=1,..., \bar{p}$, using the Elzinga-Hearn algorithm or its enhanced version as described in Elshaikh et al. [11].

(iii) Repeat steps (i) and (ii) until there is no further improvement.

b) Local search LS2

This local search is applied only when the number of facilities is p. It is based on swapping an open facility chosen randomly from the current `covering circle' with a location randomly selected from the same `covering circle'. The definition of a `covering circle' is as follows:

Let

C_1: the largest circle defined by centre X_1 and (largest) radius R_1;

C_j: the j^{th} nearest circle to the largest circle measured by the distance between the two centres where X_j and R_j define the centre and the radius of C_j, respectively; $j = 1,..., p$.

$CC_{k'}$: the area encompassed by the artificial circle centered at X_1 with a radius $\widehat{R}_{k'} = d(X_{k'}, X_1)$ if $k' > 1$, and $\widehat{R}_1 = R_1$ otherwise; $k' = 1,..., p$.

We refer to $CC_{k'}$ as the k'^{th} covering circle. In other words, this is an artificial circle with a radius defined as the distance from the centre of the largest circle X_1 to its $(k'-1)^{th}$ nearest facility defined by $X_{k'}$. The reasoning behind this idea is to concentrate the search around the neighbourhood of the largest circle as this constitutes the main characteristic of the $p-$centre problem. An example of an 8-centre problem is illustrated in Figure 1.

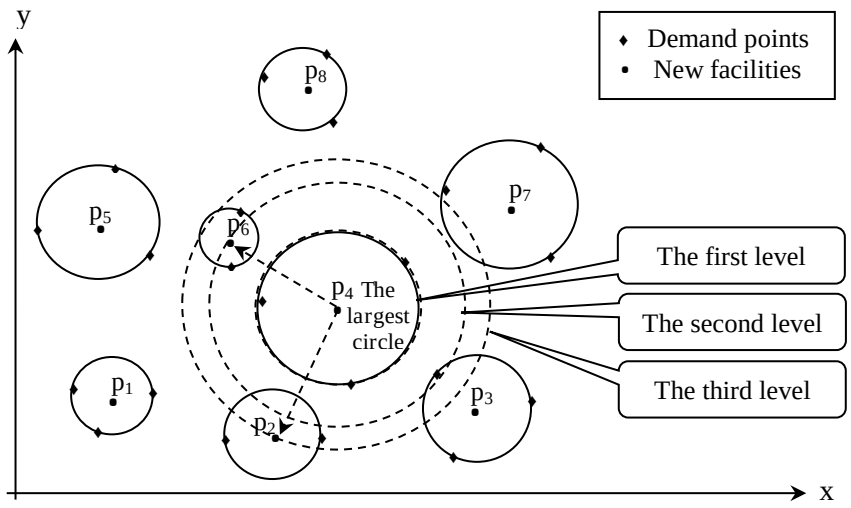

Figure 1. An example of the levels of covering circles that are dynamically increasing from the source region of an 8-centre problem

In brief, the procedure works as follows: we start from the first level ($k' = 1$) of the covering circle (the largest circle) by dropping the facility of the largest circle and inserting a facility randomly in CC_1. If the solution is not improved after applying LS1, we move to CC_2 (i.e., the second level of the covering circle by enlarging it to contain two facilities, namely, the facility of the largest circle and the nearest facility to it). One of these two candidate facilities is then randomly selected to be dropped and replaced by a location also randomly chosen in the continuous space encompassed by CC_2 followed by LS1. If the new solution is improved we revert back to level 1, where the largest circle, which may not

necessarily be the previous one, is identified again; its corresponding covering circle CC_1 is then used and the process is repeated; otherwise we continue exploring the next level. This process continues until the last level, l_{Max} say, which includes all the facilities ($l_{Max} \leq p-1$) is reached. From that point we apply a reversal move by gradually reducing the level of the covering circle until level one is reached. The swapping process is performed until no improvement is found after k_{Max} successive trials (here we set $k_{Max} = \lceil \sqrt{p} \rceil$). Note that at this point, we record the current level, $\hat{l}$ say, and the direction whether we are in the process of increasing the level (Flag = 1) or decreasing the level (Flag = -1). This is important as this information is used when we reach p again in subsequent iterations, where the search continues from the next level based on whichever level is reached at this iteration (i.e., if Flag=1 set $l = l+1$, else set $l = l-1$) while retaining the same direction. Initially, Flag is obviously set to 1 as the search starts from level 1 defining the largest circle. The steps of this procedure, which we call PROC-LS2, are given in Figure 2.

$PROC-LS2(\hat{l}, l_{Max}, k_{Max}, \hat{S}, Flag)$

(i) Let k= 0 , S= S_{best} and l= $\hat{l}$

(ii) Generate a new solution by swapping randomly one facility from S in level l, apply LS1 and set k=k+1. Let S' be the new solution

(iii) If Z(S') < Z(S), set S= S', k= 0, l=1, and go to step (ii)

(iv) If FLAG=1 then if $l < l_{Max}$, set l=l+ 1 otherwise FLAG= -1

Else if $l > 1$, set l= $l - 1$, otherwise FLAG=+1

(v) If $k < k_{Max}$, set k=k+ 1 and return to step (ii), else set $\hat{S}$ = S and $\hat{l} = l$.

Figure 2. The PROC-LS2 procedure

A brief on the three moves

Here, we briefly present the three moves.

The drop move- The strategy is to remove q facilities one by one followed each time by LS1. This process is applied when the number of open facilities is p and going down to $p-q$ (infeasible case) or starting from $p+q$ (infeasible case) and going down to p. Here, the facility chosen is the one whose removal increases the objective function the least,

which is then followed by LS1 to find a new solution with one facility less. This procedure is repeated until q facilities are removed.

The add move- Here, q facilities are inserted when the number of open facilities is p with the aim to go over the required number of facilities to $p+q$. Similarly, this is also applied when the number of open facilities reaches $p-q$.

The swap move- When the number of open facilities reaches p, we relocate randomly one open facility from the current covering circle to a point randomly chosen from the same covering circle based on the procedure LS2 (PROC-LS2) given earlier.

3.2 The GRADPERT heuristic

In [25] the added facility is chosen based on the largest cost saving among the potential facility sites as the problem is a discrete type location problem. Here, the q new facilities are added randomly one at a time in the continuous space encompassed by CC_1 instead. This solution is then examined for possible improvement by "LS1" at each of the q steps. A similar process is applied in the drop move except the removal is not performed randomly but using the least extra cost rule. The algorithm "GRADPERT" is given in Figure 3 with its main steps briefly described as follows.

Step 1

The initial solution is generated by randomly choosing p fixed points though other schemes could also be used. In our study, we chose the best solution of a multi-start with 100 runs as well as the optimal solution of the vertex p-centre problem.

Steps 2a and 3a

LS1 is used here and also in step 1 to improve upon the initial solution.

Steps 2b and 3b

When a solution with p facilities is reached, intensification is activated using LS2. Here, a swapping type process is used where one facility is chosen randomly from the covering circle (level $l=\hat{l}$) and then relocated randomly in the continuous space of the same covering circle whose size is dynamically adjusted as described previously by PROC_LS2 in Figure 2 .

Step 0: Set $q = 1$, $q_{Max} = \lceil \sqrt{p} \rceil$, k_{Max}, l_{Max} and CPU_{Max} and let $\bar{p} = p$, $\hat{l}$ =1 and Flag=1.

Step 1: Generate an initial feasible solution (S) and compute the objective function value Z(S).
Set S_{best} = S and Z_{best} = Z(S).

Step 2:

Step 2a: Perturb the solution (S_{best}) by adding one facility randomly in CC_1, apply LS1 to find the new S and set $\bar{p} = \bar{p} + 1$. ***"Perturbation via add"***

Step 2b: If $\bar{p} = p$, apply LS2 using PROC_LS2($\hat{l}$, l_{Max}, k_{Max}, $\hat{S}$, Flag) ***"Intensification Phase"***

If $Z(\hat{S}) < Z_{best}$ set $S_{best} = \hat{S}$, $Z_{best} = Z(\hat{S})$ and S= S_{best}.

Step 2c: If $\bar{p} < p + q$, go to Step 2a, else go to Step 3.

Step 3:

Step 3*a*: Perturb the solution (S_{best}) by dropping the facility that increases the objective function the least, apply LS1 to find the new S and set $\bar{p} = \bar{p} - 1$. ***"Perturbation via drop"***

Step 3b: If $\bar{p} = p$, apply LS2 using PROC_LS2($\hat{l}$, l_{Max}, k_{Max}, $\hat{S}$, Flag). ***"Intensification Phase"***

If $Z(\hat{S}) < Z_{best}$ set $S_{best} = \hat{S}$, $Z_{best} = Z(\hat{S})$ and S= S_{best}.

Step 3c: If $\bar{p} > p - q$, go to Step 3*a*,

Else

if $q < q_{Max}$ set $q = q + 1$, else set $q = 1$;

go to Step 4.

Figure 3. The GRADPERT heuristic

3.3 The STRONGPERT heuristic

It can be noted that when the solution of the p-centre location problem is not optimal, the facility serving the customers that are encompassed by the largest circle and at least one of the facilities that are around it are not in the right location. This key observation is taken into account by introducing noises around the largest circle of the current solution. To achieve this, we adopt two schemes (i) the way we add these q facilities and (ii) the way we define the destination cluster where these facilities will be sited. In (i), whenever a solution has p facilities, all the q facilities are added randomly in the continuous space encompassed by the covering circle CC_q in one step where LS1 is then activated. Similarly, q facilities are also added randomly for a solution with $p - q$ facilities to reach p facilities in one step. In (ii) we

incorporate flexibility by dynamically increasing and/or decreasing the size of the covering circle, as described in PROC-LS2.

Here, we start by adding one facility (q=1) randomly in the area encompassed by CC_1 (i.e., the largest circle). In case q=2, two new facilities are inserted randomly in the area encompassed by CC_2. This radius of the covering circle continues to increase with q until the last level is reached (i.e., $CC_{q_{Max}}$). However, in the dropping process say from $p+q$ to p and from p to $p-q$, we follow the steps of GRADPERT. The steps of STRONGPERT are therefore similar to those of "GRADPERT" of Figure 2 except Step 2a and Step 2c are modified accordingly to cater for the two schemes mentioned above.

Step 2a: Perturb the solution (S_{best}) by adding randomly one facility in the continuous space encompassed by the covering circle CC_q and set $\bar{p} = \bar{p} + 1$

Step 2c: If $\bar{p} < p + q$, repeat Step 2a, else apply LS1 and go to Step 3.

4 The Integration of Learning into the Search

In this section we incorporate learning into our perturbation-based heuristics. The aim is to identify the most promising values of q, q_{Max} and the depth of the covered area (i.e., the destination region that we insert the added facilities in).

The learning process consists of two phases. In the first phase, the information that is mentioned above is recorded during a certain time period (say for instance 25% of the total CPU time) which we call the learning phase. In the second phase, we use the obtained information about q, q_{Max} and the level of the covering circle to guide the search during the remaining time, see Figure 4 for an illustration. It is worth noting that STRONGPERT has more flexibility than GRADPERT given the size of its covering circle is dynamically changing.

Phase I: Learning process

In this phase, we record the number of times the solution is improved for each value of q (number of added/removed facilities). We also identify the minimum and the maximum q values where the latter relates to q_{Max}. In STRONGPERT, we also record the level (radius) used of the covering circle whenever the solution improves. In other words, if there is an

improvement for a given attribute (q, q_{Max} or level), the frequency of using this attribute will be increased by one.

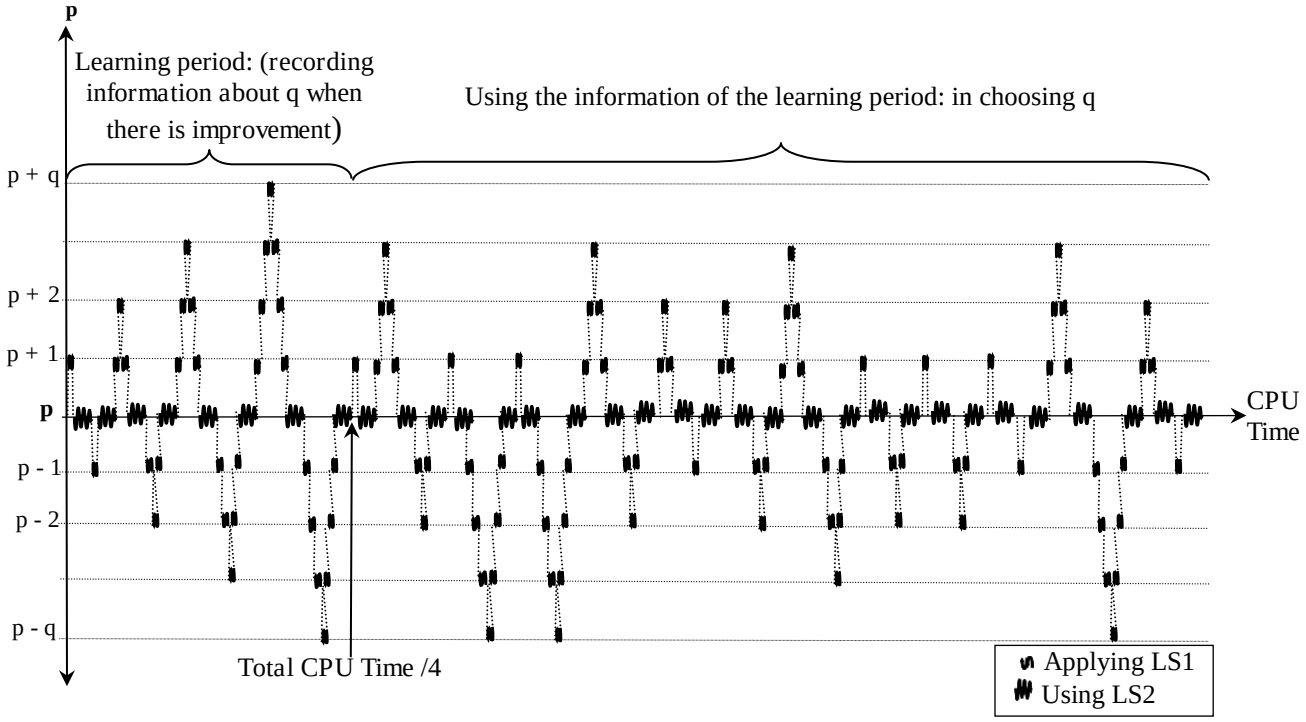

Figure 4: The GRADPERT heuristic with learning

Phase II: Integrating the information within the search

The information that is recorded in the first phase (the value of q for both schemes and the depth of the covered area in STRONGPERT) is then used to guide the search by using the following frequency of occurrence-based scheme, usually known as the inverse method. For instance, the frequency of occurrence of the q values when the solution is improved is used to compute the probabilities of occurrence of each value of q, say $P(q)$. In other words, the higher the probability of a given value of q, the higher is the chance that such a value will be chosen.

In brief, the idea is to choose $\alpha \in [0,1]$ randomly and compute $\hat{q} = F^{-1}(\alpha)$ with $F(q) = \sum_{t=1}^{q} P(t)$ being the cumulative probability distribution, and $P(t)$ refers to the probability of choosing the t^{th} q value $(t = 1, ..., q_{Max})$. The same calculations are performed for the other attributes.

5 Computational Results

The perturbation-based heuristics are coded in C++ and executed on a laptop computer with an Intel Core 2 Duo processor, 2.0 GHz CPU and 4G memory. For the vertex p-centre problem, the IBM ILOG CPLEX12.5 Concert library is used. The proposed heuristics are tested on TSP-Lib data sets (n=439, 575, 783, 1002 and 1323) using values of p ranging from p=10 to 100 with an increment of 10.

To be consistent with previous results given in [11], we also used the CPU times corresponding to 10,000 iterations of the multi-start as our stopping criterion. The effect of this stopping rule on the convergence of the proposed perturbation heuristics will be briefly examined at the end of this section (see subsection 5.3). We compute the deviation from the best solution as Deviation (%) = $\frac{(Z_H - Z_{best})}{Z_{best}}.100$ with Z_H denoting the objective value found by heuristic 'H', and Z_{best} being the optimal or the best value found over all the heuristics.

We propose two strategies for generating the initial solution:

a) The solution of the multi-start procedure with 100 runs.
b) The optimal solution of the vertex p-centre problem.

In (b), we adopt the set covering-based approach based on Salhi and Al-Khedhairi [26] that uses the efficient exact method of Al-Khedhairi and Salhi [1] with tight upper and lower bounds at the initialisation phase of the binary search. For convenience and benchmarking purposes, we also report for the first time the optimal solutions of the discrete problems for all five data sets using $p = 10,...,100$ including their corresponding CPU times (in secs), see Appendix A.

5.1 Initial Observations

The detailed results of GRADPERT and STRONGPERT with and without learning using strategies (a) and (b) as well as the optimal discrete solutions and their corresponding continuous solutions are given in Appendix B. In general, it was found that incorporating learning within the search has enhanced the efficiency of the perturbation-based heuristics.

The continuous solution obtained from the optimal discrete solution improves the objective value up to approximately 10% (when n = 575 and p =80 and 100), with an average of over

3.5%. This result is also highlighted by Hansen and Mladenović [16] and Gamal and Salhi [14] for the multi-source Weber problem. It should be noted, however, that using the best solution of the multi-start procedure with 100 runs as an initial solution (i.e., strategy (a)), though not initially as competitive as the optimal discrete solution (i.e., strategy (b)), yields in most cases better overall results when using the perturbation heuristics (i.e., the final result). This is due to the excessive time used for the optimal method at the discrete phase leaving just a relatively small time if any for the perturbation method to improve upon the solution. For example when $n = 783$ and p = 40, 50 and 60, there was no remaining time at all to run the perturbation-based heuristic as the CPU time corresponding to the 10,000 runs of the multi-start was even smaller than the time used to find the optimal discrete solution (see Appendices A and B).

In subsequent comparisons, we will therefore concentrate and report the results of both perturbation methods with learning incorporated and with strategy (a) only for the generation of the initial solution.

5.2 Comparison against other methods

For the smaller instances (i.e., $n = 439$ and $n = 575$), we used the optimal solutions for comparison purposes. We also implemented the optimal method of Drezner [6] where we obtained the optimal solutions for both $n = 439$ and $n = 575$ though the CPU time was excessively large, sometimes in excess of 24 hours, especially for $n = 575$ and small values of p, see Table 1 for the summary results. It is worth noting that this is the first time that the optimal solutions for $n = 575$ are reported. The optimal solutions for $n = 439$ were previously found by Chen and Chen [3] using the best of their relaxation methods.

As no optimal solutions are available for the rest of the data sets (783, 1002 and 1323), we have implemented in C++ those classical heuristics that were briefly reviewed in the introduction section. These include the composite heuristic H2 given by Drezner [6] and the drop-based method (STEPDOWN) of Eiselt and Charlesworth [13]. In our experiments, we tested the STEPDOWN method with $\beta = 0.5$ and 1, but we report the results of the best variant namely when $\beta = 0.5$ with and without LS1. This observation was also noted in [13]. For completeness, we have also added the recently published results by the two best variants of VNS given in Elshaikh et al. [11].

Table 1: Comparison vs other heuristics (Deviation %)

n	p	Overall best or optimal solutions (Z)	Initial solution based on 100 restarts		Multi-Start (H2)[a]	STEPDOWN[b] (β = 0.5)		Initial solution based on 100 restarts	
			VNS(CN)	VNS(FN) with memory (VNS-M)		Without LS1	with LS1	GRADPERT	STRONGPERT
439	10	1716.5099**	**0**	**0**	1.451	2.131	3.904	**0**	**0**
	20	1029.7148**	**0**	**0**	9.422	9.989	8.801	**0**	**0**
	30	739.1929**	**0**	**0**	31.901	**0**	14.054	**0**	**0**
	40	580.0054**	**0**	**0**	17.964	9.489	9.489	**0**	**0**
	50	468.5416**	**0.674**	**0.674**	32.697	21.673	16.228	0.850	1.184
	60	400.1952**	**0.349**	**0.349**	25.017	20.203	9.856	**0.349**	**0.349**
	70	357.9455**	1.272	**0**	34.391	12.835	9.098	1.230	**0**
	80	312.5000**	1.203	**0.017**	26.301	17.644	16.276	0.956	0.956
	90	280.9025**	0.395	0.395	35.310	16.466	12.047	**0**	0.395
	100	256.680194**	0.395	0.896	17.282	13.271	9.869	1.353	**0.347**
	Average	614.219	0.429	0.233	23.174	12.370	10.962	0.474	0.323
575	10	67.9258*	0.998	**0**	1.910	4.121	3.266	0.998	1.910
	20	45.4750*	**0.323**	**0.323**	4.661	9.323	14.881	**0.323**	**0.323**
	30	35.5563*	**0**	0.156	11.527	21.139	8.607	0.670	1.724
	40	30.0633*	2.327	**0.67**	10.984	12.124	15.275	1.408	1.166
	50	25.8263*	**1.713**	2.489	15.459	13.238	14.372	3.264	3.671
	60	23.1625*	4.549	2.28	17.021	20.460	16.815	1.542	**1.181**
	70	20.8581*	3.105	**0.961**	21.382	17.558	17.684	2.406	1.731
	80	19.0263*	2.964	4.326	23.037	23.541	25.290	**2.792**	5.118
	90	17.4604*	3.487	**2.652**	26.422	17.513	16.966	3.250	4.751
	100	16.4200*	1.771	**1.695**	23.653	25.440	17.502	2.135	3.577
	Average	30.1774	2.124	1.555	15.606	16.446	15.066	1.879	2.515
783	10	79.313	**0**	**0**	0.844	10.702	9.347	**0**	**0**
	20	53.441+	0.466	**0.037**	3.892	9.959	5.217	2.652	2.171
	30	42.395	**0**	0.494	8.128	18.139	11.369	5.608	1.365
	40	35.962+	1.591	**0.411**	10.006	9.287	8.216	0.999	1.663
	50	31.184+	0.911	0.72	12.660	17.999	12.008	3.913	**0.553**
	60	28.053	**0**	1.098	17.599	21.093	20.369	2.871	0.334
	70	25.446	**0**	0.694	17.192	21.892	16.714	1.46	2.161
	80	23.560	0.845	**0**	18.921	11.109	21.432	2.418	0.443
	90	21.710	1.572	**0**	14.994	19.517	10.788	2.92	2.931
	100	20.334	1.086	**0**	18.259	18.259	16.508	1.086	2.335
	Average	36.140	0.647	**0.345**	12.250	15.796	13.197	2.393	1.396
1002	10	2389.360	**0**	**0**	0	1.500	8.565	**0**	**0**
	20	1607.530	0.125	1.416	2.488	9.538	9.541	0.125	**0**
	30	1231.360	0.108	**0**	9.145	12.126	17.352	**0**	**0**
	40	1021.410	2.19	0.88	13.952	12.323	11.869	**0**	**0**
	50	901.455++	0.529	0.724	15.050	16.412	14.110	**0.185**	0.216
	60	795.709	0.879	0.725	17.389	9.081	10.681	**0**	0.588
	70	725.431	1.216	0.238	15.864	17.070	16.613	0.144	**0**
	80	660.019	2.458	1.778	15.945	27.328	17.482	**0**	0.135
	90	604.152+	**0.057**	0.802	24.141	17.041	28.346	0.802	1.022
	100	559.017+	2.078	2.078	24.592	16.624	16.624	2.078	**1.242**
	Average	1049.544	0.964	0.864	13.857	13.904	15.118	0.333	**0.320**
1323	10	2897.490+	0.237	**0.067**	0.328	5.177	2.394	**0.067**	**0.067**
	20	1868.920++	**0.958**	**0.958**	5.298	10.438	14.724	**0.958**	1.151
	30	1466.970+	1.622	**0.984**	6.368	14.554	14.554	1.743	1.614
	40	1236.380	**0**	1.21	12.177	18.038	19.269	0.73	1.045
	50	1060.820	**0**	0.42	15.378	17.216	17.216	0.681	0.681
	60	940.691	1.354	0.125	13.719	12.415	17.655	0.102	**0**
	70	844.967	0.934	**0**	15.413	14.942	16.919	1.306	2.048
	80	774.764	1.092	**0**	16.897	21.416	21.416	1.092	1.823
	90	719.580	0.807	2.265	20.076	15.880	30.728	2.265	**0**
	100	662.936+	2.237	5.129	23.396	23.268	17.372	0.785	**0.015**
	Average	1247.352	0.924	1.116	12.905	15.334	17.225	0.973	**0.844**
Overall Average		595.486	1.018	0.823	15.558	14.770	14.314	1.210	1.080
# best			18	27	1	1	0	17	21

a: Drezner (H2) algorithm [6]. **b:** Eiselt and Charlesworth (STEPDOWN) algorithm [13].
****:** Optimal solution found by Drezner (exact) algorithm [6] and Chen and Chen [3].
***:** Optimal solution found by Drezner (exact) algorithm [6].
+: Found by other variants in Elshaikh et al. [11]. **++:** Found by other variants shown in Appendix B.

It can be noted that the performance of STRONGPERT was slightly superior to its counterpart GRADPERT. The overall average deviation values from the optimal (or the best) solutions over all the instances is 0.844% and 0.973% respectively. In addition, both perturbation methods found 5 optimal solutions for $n = 439$. When compared against other heuristics, our perturbation methods perform really well especially in the two largest instances (n=1002; 1323) where average deviations of 0.3 and 0.8% were recorded for STRONGPERT and GRADPERT respectively. Also, in general, the proposed perturbation heuristics behave comparably well against the recent VNS based metaheuristics (Elshaikh et al. [11]) while producing over 40% best solutions (21 out of 50). As expected the constructive heuristics such as H2 and the best variant of STEPDOWN were not as competitive though the latter used less computational time compared to the others whose maximum time was set by the time of the 10,000 runs of the multi-start (H1). The CPU times of these methods are reported in Table 2. Note that the CPU time for H2, VNS and the perturbation-based methods are not given as these are the same as the ones used by H1.

5.3 Convergence behaviour of the perturbation heuristics

As mentioned earlier, we used as our stopping criterion the cpu time required to perform 10,000 iterations of the multi-start (H1). This is pursued for consistency reasons as previous results are also based on the same criterion, see [11]. However, the perturbation heuristics seem to converge much earlier if other stopping rules were used instead, say if the search terminates when there is no improvement after a certain number of cycles. For illustration purposes, we used an instance with ($n = 439$) and GRADPERT as the perturbation method. Here, we record the number of improvements, the cumulative gap in % at each improvement from the final solution, and also the time spent in % until the last improvement is realised. We identified two classes with class I using $p \le 60$ and class II for the rest. It was found that for class I, GRADPERT required approximately 12-15 improvements to reach the optimal (best) solution while consuming a tiny fraction of the total time only (1-2%) with the exception of p=50 where 6% of the time was needed. For class II (i.e., larger values of p), approximately 30 improvements were needed accounting for only 20 to 30% of the total time to achieve the best solution. Two graphs, representing both classes, are given in Figure 5 that show the patterns in terms of solution gap from the best (in %) and the % time required from the total time for the case of p=40 (class I) and p=80 (class II). Similar patterns were also observed for the other instances. This result demonstrates that the proposed perturbation

heuristic is rather fast at obtaining good quality solutions if other stopping rules, as mentioned earlier, are used instead.

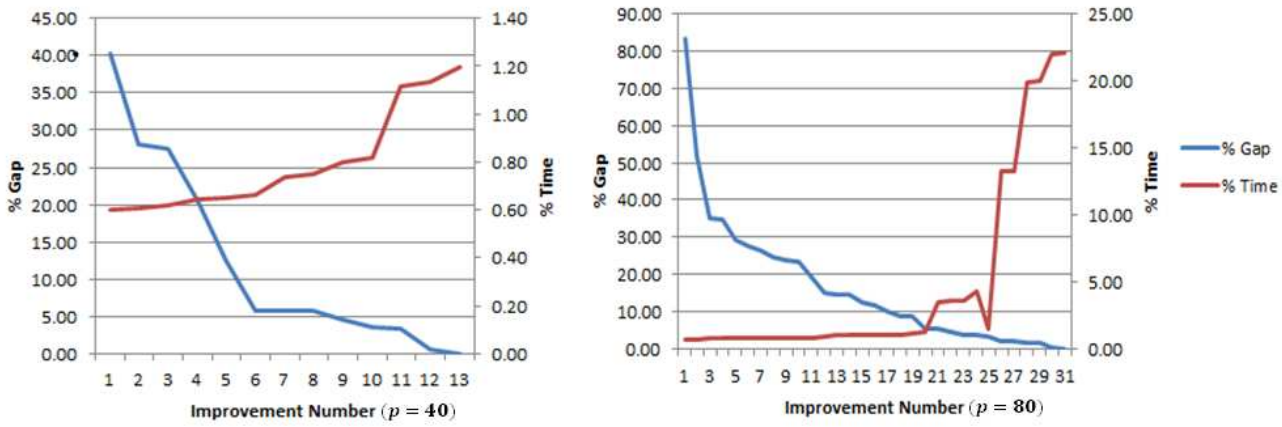

Figure 5: Typical convergence patterns of the perturbation heuristics:
Case of GRADPERT with $n = 439$ for class I ($p \leq 60$) and class II ($p \geq 70$)

6 Conclusion and Suggestions

A new perturbation-based heuristic is designed to solve the continuous p-centre problem. The idea is to allow the number of facilities to be higher and lower than p in order to act as a filtering process where the promising facility locations tend to stay in the chosen set. We also guide the search by allowing the amount of perturbation to vary adaptively instead of being a constant throughout the search as originally proposed in the literature. A novel local search that uses the concept of covering circles that are dynamically adjusted is also developed and the incorporation of learning is taken into account to guide the search. The obtained results are encouraging when tested on several TSP-Lib data sets ($n = 439, 575, 783, 1002$ and 1323) using various values of p . The proposed perturbation heuristic significantly outperforms some known composite heuristics and obtains comparable results when compared to those powerful metaheuristics recently given in Elshaikh et al. [11]. We also record for the first time optimal solutions for the case of $n = 575$ for all values of p using the optimal method of Drezner [6] though the computational burden for this data set was found to be relatively high. As by product of this work, we also report for the first time the optimal solution of the vertex p centre problem for these data sets.

Table 2: Total CPU time (in secs) of Multi-Start (H1), Drezner's optimal method and the drop-based method (STEPDOWN) and # iterations of H2

n	P	CPU time (secs)				# iterations
		H1 Total CPU time (10000 runs)	Exact algorithm of Drezner [6]	STEPDOWN[b] (β=0.5)		$H2^{+}$
				without LS1	with LS1	
439	10	435.058	6252.720	7.980	74.140	3784
	20	751.331	56753.000	9.375	71.977	3875
	30	1018.990	37017.100	7.956	84.859	4557
	40	1171.130	31355.000	8.034	85.985	4962
	50	1730.060	4939.250	8.064	84.002	6958
	60	1984.200	4956.450	9.976	82.079	7871
	70	2087.360	3170.890	8.886	84.791	7904
	80	1943.060	2186.270	9.003	84.269	6718
	90	1988.140	1258.220	11.174	84.901	7491
	100	1866.300	462.297	10.236	93.691	6228
	Average	1497.563	**14835.100**	9.068	83.069	6034.800
575	10	541.690	83898.600	15.865	158.461	4931
	20	943.640	19087.600	15.523	157.174	5293
	30	1190.150	9743.910	15.444	156.848	5011
	40	1436.050	41733.000	15.726	162.909	4786
	50	1664.060	9612.610	20.078	168.853	4454
	60	1789.300	28344.000	18.972	162.718	3989
	70	2143.630	40256.900	15.616	162.793	3478
	80	2167.660	40181.700	19.945	148.865	3528
	90	2307.930	4260.100	20.001	151.953	3891
	100	2531.670	33694.000	16.294	153.133	3914
	Average	1671.578	**31081.242**	17.346	158.371	4327.500
783	10	909.638	N/A	39.027	397.385	6697
	20	1555.440	N/A	40.342	404.673	6422
	30	2055.590	N/A	42.436	445.683	5534
	40	2403.090	N/A	39.673	451.611	4540
	50	2514.470	N/A	39.153	403.104	4328
	60	2842.810	N/A	40.304	412.328	4370
	70	3154.780	N/A	38.877	383.989	4555
	80	4466.130	N/A	38.719	403.217	5054
	90	3646.990	N/A	50.667	379.012	4156
	100	4075.510	N/A	39.076	377.540	4225
	Average	2762.446	N/A	40.827	405.854	4988.100
1002	10	947.822	N/A	77.863	812.409	5484
	20	1627.170	N/A	77.800	887.473	4837
	30	2562.060	N/A	78.449	863.260	4938
	40	2959.210	N/A	77.440	803.861	4214
	50	3682.880	N/A	78.356	827.421	5778
	60	4520.700	N/A	98.117	798.079	6821
	70	6640.190	N/A	78.648	849.338	8506
	80	7026.190	N/A	97.464	826.794	8142
	90	6883.150	N/A	78.142	831.767	6393
	100	7131.510	N/A	79.835	900.827	6131
	Average	4398.088	N/A	82.211	840.123	6124.400
1323	10	1584.920	N/A	175.984	1982.580	5699
	20	2439.180	N/A	174.922	1915.420	5189
	30	3454.570	N/A	178.851	1787.050	4767
	40	4093.150	N/A	177.178	1849.580	4443
	50	5677.000	N/A	169.932	1839.270	5586
	60	6527.830	N/A	217.186	2068.800	5689
	70	7515.280	N/A	169.980	1824.060	5467
	80	7905.080	N/A	175.265	2324.090	4771
	90	8417.760	N/A	174.331	2169.950	4568
	100	9015.060	N/A	187.385	2315.050	4403
	Average	5662.983	N/A	180.101	2007.585	5058.200

+: CPU of H2 is not reported as it is the same as H1. * best result found by STEPDOWN

For future research, the way the q facilities are inserted or dropped could be revisited. For instance, when the number of facilities reaches p a stronger local search than LS2, or even a short meta-heuristic including variable neighbourhood descent (VND), could also be introduced to form a powerful hybrid. Alternatively, a perturbation-based heuristic may be considered as a new local search within meta-heuristic frameworks such as variable neighbourhood search. Other stopping rules could be worth exploring so as to terminate the search earlier if necessary as highlighted in subsection 5.3. The perturbation scheme allows us to generate many candidate "centres" during its up and down trajectories. These new centres could be used within the new reformulation local search (RLS) framework (see Brimberg et al. [2]) to increase the set of potential sites, and hence, improve the ability to find better solutions in the continuous space. Hybrid algorithms based on perturbation methods and RLS could be considered in future for the continuous p-centre problem in particular and other related continuous location problems in general. The proposed perturbation methodology can be extended to very large continuous problems where little work has been done as highlighted in Irawan and Salhi [17]. We also believe that the optimal method of Drezner [6] and the relaxation-based technique of Chen and Chen [3], both have scope for improving their implementations, and hence, could be worth revisiting.

Acknowledgments- We would like to thank the referees and the area editor for their constructive comments that improved both the content as well as the presentation of the paper. The first author is also grateful to the University of Misurata for his PhD studentship.

References

[1] Al-Khedhairi A and Salhi S. Enhancements to two exact algorithms for solving the vertex p-center problem, Journal of Mathematical Modelling Algorithms 2005; 4: 129-147.

[2] Brimberg J, Drezner Z, Mladenović N and Salhi S. A new local search for continuous location problems, European Journal of Operational Research; 2014: 232: 256-265.

[3] Chen D and Chen R. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. Computers & Operations Research 2009; 36: 1646-1655.

[4] Chen D and Handler GY. Relaxation method for the solution of the minimax location-allocation problem in Euclidean space. Naval Research Logistics 1987; 34: 775-787.

[5] Cooper L. Heuristic methods for location-allocation problem, SIAM Review 1964; 6: 37-53.

[6] Drezner Z. The p-center problem- heuristic and optimal algorithms. Journal of the Operational Research Society 1984; 35: 741-748.

[7] Drezner Z. The planar two-center and two-median problems. Transportation Science 1984; 18: 351-361.

[8] Drezner Z and Suzuki A. The p-Center location problem in an area. Location Science 1996; 4: 69-82.

[9] Drezner Z. Continuous Center Problems. In H.A. Eiselt and V. Marionov (eds). Foundations of Location Analysis, Springer-Verlag, pp 63-78, 2011.

[10] Elshaikh A. Adaptive heuristic methods for the continuous p-centre location problem, PhD Dissertation 2014; Appendix C, University of Kent, Canterbury, UK.

[11] Elshaikh A, Salhi S and Nagy G. The continuous p-centre problem: An investigation into variable neighbourhood search with memory, European Journal of Operational Research 2015; 241: 606-621.

[12] Elzinga J and Hearn DW. Geometrical solutions for some minimax location problems, Transportation Science 1972; 6: 379-394.

[13] Eiselt HA and Charlesworth GA. A note on p-center problems in the plane. Transportation Science 1986; 20: 130-133.

[14] Gamal MDH and Salhi S. Constructive heuristics for the uncapacitated continuous location-allocation problem, Journal of the Operational Research Society 2001; 52: 821-829.

[15] Hanafi S and Freville A. An efficient tabu search approach for the 0-1 Multi-dimensional knapsack problem, European Journal of Operational Research 1998; 106: 659-675.

[16] Hansen P, Mladenović N and Taillard E. Heuristic solution of the multisource Weber problem as a p-median problem. Operations Research Letters 1998; 22: 55-62.

[17] Irawan CA and Salhi S. Aggregation and non aggregation techniques for large facility location problems- A survey. Yugoslav Journal of Operations Research 2015; 35:312-341.

[18] Kavah, A, and Nasr, H, (2011), Solving the conditional and unconditional p-centre problem with modified harmony search: A real case study, Scientia Iranica, 4, pp. 867-877.

[19] Lu, C., (2013), Robust weighted vertex p −center model considering uncertain data: An application to emergency management, European Journal of Operational Research, 230, pp.113-121.

[20] Megiddo N and Supowit KJ. On the complexity of some common geometric location problems. SIAM Journal of Computing, 13, 182-196, 1984.

[21] Murray AT and Wei R. A computational approach for eliminating error in the solution of the location set covering problem. Computers & Operations Research 2013; 224: 52-64.

[22] Pacheco, J. A & Casado, S, (2004), Solving two location models with few facilities by using a hybrid heuristic: a real health resources case, Computers and Operations Research, 32, pp. 3075-3091.

[23] Plastria F. Continuous covering location problem, Facility location: applications and theory (Z. Drezner ed.), New York: Springer; 2002. p.37–79.

[24] Richard, D, Beguin, H & Peeters, D, (1990), The location of fire stations in a rural environment: a case study, Environment and Planning, 22, pp. 39-52.

[25] Salhi, S. A Perturbation Heuristic for a Class of Location Problems, Journal of the Operational Research Society 1997; 48: 1233-1240.

[26] Salhi S and Al-Khedhairi A. Integrating heuristic information into exact methods: The case of the vertex p-centre problem. Journal of the Operational Research Society 2010; 61: 1619-1631.

[27] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming – Lecture Notes in Computer Science 1998; 1520: 417-431.

[28] Suzuki A and Okabe A. Using Voronoi diagrams, Facility Location: A Survey of Applications and Methods, (Z. Drezner ed.), New York: Springer; 1995. p. 103–118.

[29] Teitz M and Bart P. Heuristic Methods for Estimating the General Vertex Median of a Weighted Graph, Operations Research. 16: 955-961, 1968.

[30] Wei H, Murray AT and Xiao N. Solving the continuous space p-centre problem: planning application issues, IMA Journal of Management Mathematics 2006; 17: 413–425.

[31] Xu S, Freund R and Sun J. Solution methodologies for the smallest enclosing circle problem. Computational Optimization and Applications. 25: 283–292, 2003.

[32] Zainuddin ZM and Salhi S. A Perturbation-Based Heuristic for the Capacitated Multisource Weber Problem, European Journal of Operational Research 2007; 179: 1194-1207.

Appendix A: Optimal solutions and CPU times (secs) for the vertex p-centre problem (n=439,…,1323)

n \ p	439		575		783		1002		1323	
	Z	CPU	Z	CPU	Z	CPU	Z	CPU	Z	CPU
10	1971.832	3.389	72.670	15.520	83.486	10.132	2540.177	12.833	3077.297	40.498
20	1185.59	4.239	49.244	37.954	56.850	195.212	1726.267	44.577	2016.396	75.709
30	883.529	3.959	39.408	408.546	46.065	577.578	1346.291	22.525	1631.501	90.568
40	671.751	4.118	33.301	224.848	39.560	7555.408	1171.537	108.316	1352.36	203.265
50	564.025	3.863	29.427	286.438	34.785	4907.306	1029.563	44.606	1187.265	320.32
60	500	4.494	27	165.433	31.400	8324.649	912.414	9.944	1063.014	515.759
70	474.341	4.003	24.758	142.520	28.844	2204.851	850	11.063	971.925	110.694
80	412.310	4.140	23.345	88.325	26.925	1470.533	761.577	5.395	895.055	83.543
90	395.284	3.171	21.931	53.012	25.495	573.266	715.891	5.307	832	53.821
100	350	4.444	20.615	22.490	24.041	144.354	670.82	7.291	787.095	50.091
Average	740.866	3.982	34.170	144.507	39.745	2596.329	1172.454	27.1857	1381.391	154.427

Appendix B: Deviation (%) from best of Multi-Start, GRADPERT and STRONGPERT (with and without learning)

n	p	Overall Best (Z)	Multi-Start (10 000 Runs)	Initial solution with the multi-start algorithm with 100 runs				Optimal Discrete solutions	Optimal Discrete Solutions + Continuous	Initial solution using optimal discrete solutions			
				GRADPERT		STRONGPERT				GRADPERT		STRONGPERT	
				No Learning	With Learning	No Learning	With Learning			No Learning	With Learning	No Learning	With Learning
575	10	67.926	1.910	0.998	0.998	0.998	1.910	6.984	4.984	**0**	**0**	0.998	**0**
	20	45.622	3.097	**0**	**0**	0.882	**0**	7.939	3.745	**0**	1.025	1.025	**0**
	30	35.556	9.050	1.523	0.670	1.534	1.724	10.833	5.813	0.503	0.959	0.503	**0**
	40	30.414	13.946	1.954	0.240	0.036	**0**	9.493	5.792	1.462	1.617	1.617	0.942
	50	26.319	17.185	2.413	1.332	1.829	1.731	11.810	7.911	1.332	1.189	0.919	**0**
	60	23.436	19.645	0.544	0.357	2.095	**0**	15.207	11.339	4.365	3.596	3.309	4.561
	70	21.219	13.888	1.020	0.664	1.020	**0**	16.678	11.641	0.195	0.818	1.939	1.939
	80	19.266	26.850	1.761	1.515	4.344	3.811	21.173	11.485	2.580	**0**	1.515	3.266
	90	17.805	24.391	2.107	1.254	1.487	2.726	23.177	15.377	**0**	2.292	1.487	0.913
	100	16.711	27.822	0.948	0.357	0.580	1.775	23.363	13.861	0.580	0.028	**0**	2.214
	Average	30.427	15.778	1.327	**0.739**	1.481	1.368	14.666	9.1946	1.102	1.152	1.331	1.384
783	10	79.313	**0**	**0**	**0**	**0**	**0**	5.262	4.557	**0**	**0**	**0**	**0**
	20	53.690	2.2749	2.38	2.176	0.582	1.697	5.886	5.665	**0**	0.424	0.582	0.977
	30	42.801	10.776	1.038	4.606	3.398	0.404	7.627	3.181	**0**	1.951	0.125	1.009
	40	36.321	9.656	0.586	**0**	0.417	0.657	10.264+	10.264+	10.264+	10.264+	10.264+	10.264+
	50	31.357	15.361	**0**	3.341	2.102	**0**	11.253+	11.253+	11.253+	11.253+	11.253+	11.253+
	60	28.128	17.871	0.866	2.597	**0**	0.066	23.218+	23.218+	23.218+	23.218+	23.218+	23.218+
	70	25.446	20.885	**0**	1.46	1.667	2.161	13.356	9.633	4.218	4.391	5.034	3.145
	80	23.665	22.127	0.646	1.967	0.800	**0**	13.778	10.234	1.109	0.800	0.893	0.797
	90	21.759	24.426	**0**	2.688	0.898	2.698	17.169	11.465	1.576	1.114	2.382	3.813
	100	20.334	26.014	1.086	1.086	0.470	2.335	18.231	17.129	**0**	0.058	1.056	1.276
	Average	36.281	14.939	**0.66**	1.992	1.033	1.002	11.279	10.6599	5.164	5.347	5.481	5.575
1002	10	2389.360	0.889	**0**	**0**	**0**	**0**	6.312	4.031	1.102	**0**	1.102	**0**
	20	1607.530	4.792	0.125	0.125	0.125	**0**	7.386	4.325	1.588	1.588	1.588	1.176
	30	1231.360	8.418	0.494	**0**	1.485	**0**	9.334	5.368	1.092	1.180	0.658	0.658
	40	1021.410	18.095	2.244	**0**	2.244	**0**	14.698	6.813	2.244	2.044	2.244	2.044
	50	901.455	17.005	1.822	0.185	1.753	0.216	14.211	11.537	0.529	**0**	1.244	0.529
	60	795.709	22.007	2.571	**0**	0.725	0.588	14.667	12.406	4.622	2.479	2.936	3.661
	70	725.431	17.980	1.827	0.144	0.238	**0**	17.172	11.351	1.216	0.223	0.178	0.178
	80	660.019	22.913	0.135	**0**	1.989	0.135	15.387	9.845	0.644	0.135	1.778	1.778
	90	604.494	28.273	0.745	0.745	0.745	0.965	18.428	12.503	0.154	0.745	0.745	**0**
	100	559.061	29.415	1.026	2.07	2.948	1.234	19.990	14.097	1.026	3.722	2.070	**0**
	Average	1049.583	16.979	1.099	0.327	1.225	**0.314**	13.759	9.228	1.422	1.212	1.454	1.002
1323	10	2899.420	0.260	0.084	**0**	0.170	**0**	6.135	1.737	0.170	0.414	0.170	0.414
	20	1868.920	5.414	**0**	0.958	0.958	1.151	7.891	5.604	0.958	0.958	0.958	0.958
	30	1477.590	7.514	**0**	1.012	1.463	0.883	10.416	6.850	0.869	1.192	1.192	1.192
	40	1240.620	12.021	1.793	0.386	0.386	0.700	9.007	4.627	**0**	**0**	**0**	**0**
	50	1061.660	15.897	0.601	0.601	1.444	0.601	11.831	7.930	**0**	**0**	0.621	**0**
	60	940.691	12.804	0.102	0.102	0.104	**0**	13.004	11.561	1.532	0.198	1.734	0.403
	70	849.782	18.706	0.858	0.732	2.965	1.469	14.373	10.698	**0**	**0**	0.732	0.444
	80	776.401	15.092	1.512	0.879	0.879	1.608	15.283	11.510	1.608	2.189	3.675	**0**
	90	719.580	24.180	1.631	2.265	0.380	**0**	15.623	14.425	0.486	1.643	0.373	1.631
	100	663.035	28.614	2.189	0.770	1.975	**0**	18.711	12.908	1.564	2.517	1.992	2.221
	Average	1249.770	14.05	0.877	0.77	1.072	**0.641**	12.227	8.785	0.719	0.911	1.145	0.726
Overall Average		591.515	15.437	0.991	0.957	1.203	**0.831**	12.983	9.467	2.102	2.156	2.353	2.172
# best			1	8	9	3	**16**	0	0	10	8	3	11

Bold: The best solutions found.

+: Optimal discrete not guaranteed due to CPU time needed being larger than the maximum allowed CPU time.