Optimal Solutions for the Continuous $p-$Centre Problem and Related $\alpha-$Neighbour and Conditional Problems: A Relaxation-Based Algorithm*

[a]Becky Callaghan, [a]Said Salhi, [b]Jack Brimberg

[a]*Centre for Logistics & Heuristic Optimisation (CLHO), Kent Business School, The University of Kent, Canterbury, Kent, UK, CT2 7NZ.*
[b]*Department of Mathematics and Computer Science, Royal Military College of Canada, Kingston, ON, Canada K7K 7B4*

Abstract

This paper aims to solve large continuous $p-$centre problems optimally by re-examining a recent relaxation-based algorithm. The algorithm is strengthened by adding four mathematically supported enhancements to improve its efficiency. This revised relaxation algorithm yields a massive reduction in computational time enabling for the first time larger data sets to be solved optimally (e.g. up to 1323 nodes). The enhanced algorithm is also shown to be flexible as it can be easily adapted to optimally solve related practical location problems that are frequently faced by senior management when making strategic decisions. These include the $\alpha-$neighbour $p-$centre problem and the conditional $p-$centre problem. A scenario analysis using variable α is also performed to provide further managerial insights.

Keywords: Location, $p-$centre problem, $\alpha-$neighbourhood, conditional, continuous space, relaxation method, optimal solutions, managerial insights.

1. Introduction

Finding the best location for a facility can be a challenge, especially for public service facilities such as hospitals, fire stations and ambulance stations. For example, if an ambulance is summoned for someone having a heart attack, both the time and distance from the ambu-

*This research has been supported in part by the UK Research Council EPSRC EP/L504981/1 and the Spanish Ministry of Economy & Competitiveness, research project MTH2015.70260-P, in part financed by the European regional Development Fund (ERDF).

Email addresses: bc349@kent.ac.uk ([a]Becky Callaghan), s.salhi@kent.ac.uk ([a]Said Salhi), Jack.Brimbreg@rmc.ca ([b]Jack Brimberg)

lance station to the patient could mean the difference between life and death. This dilemma of locating emergency facilities efficiently has led to a large number of studies where various mathematical models and algorithms have been suggested. The location problem that this paper will focus on is the continuous $p-$centre problem, alongside two related problems known as the $\alpha-$neighbour and the conditional $p-$centre problem.

The $p-$centre problem wishes to locate p facilities on a plane or network to serve n demand points such that the maximum distance, time or cost is minimised. In other words, it aims to respond to the worst case scenario. The problem can be categorised as discrete or continuous, but this paper will be focusing on the latter. In other words, the facilities can be located anywhere in the plane, rather than at prespecified locations as in the discrete case. Note that the continuous $p-$centre problem can provide useful greenfield solutions for the discrete $p-$centre problem or promising regions of a network where potential sites should be chosen. For more information on the discrete case, see Chen & Chen (2009), Irawan *et al* (2016), and references therein.

Traditionally, the continuous $p-$centre problem is formulated as the Euclidean unweighted $p-$centre problem, though a non-linear mathematical programming formulation does also exist. However, there is a binary linear programming formulation, as defined by CP_1, with the centres of all critical circles making up the set of potential facility sites. This finite dominating set is found by constructing circles from one, two or three demand points, see Chen & Handler (1987). This limits the total number of potential facility sites to be at most $m = \binom{n}{3} + \binom{n}{2} + n$, made up of n null circles based on one point, $\binom{n}{2}$ circles defined from two points and $\binom{n}{3}$ circles defined from three points. Note that m could be relatively smaller due to the geometry of the three critical points.

$$CP_1 : \text{Minimise} \quad Z \tag{1}$$

$$\text{subject to} \quad \sum_{j \in J} A_{i,j} x_j \geq 1 \quad \forall i \in I, \tag{2}$$

$$\sum_{j \in J} x_j = p, \tag{3}$$

$$Z \geq x_j r_j \quad \forall j \in J, \tag{4}$$

$$x_j \in \{0, 1\}, \ \forall j \in J. \tag{5}$$

where

Z: the maximum distance between a facility and a covered demand point,

I: set of demand points indexed by $i = 1 \ldots n$,

J: set of identified candidate circles indexed by $j = 1 \ldots m$ (i.e. their known centres act as potential sites for facilities),

r_j: the radius of circle C_j, $j \in J$,

$d_{i,j}$: Euclidean distance from demand point i to the centre of circle C_j, $i \in I, j \in J$;

p: number of facilities to locate,

$$A_{i,j} = \begin{cases} 1 \text{ if } d_{i,j} \leq r_j \text{ (i.e. demand point } i \text{ will be covered by a facility located at the centre of circle } C_j) \text{ ,} \\ 0 \text{ else, } i \in I, j \in J. \end{cases}$$

$$x_j = \begin{cases} 1 \text{ if circle } C_j \text{ is selected} \\ 0 \text{ else, } j \in J. \end{cases}$$

The objective function (1) refers to minimising the maximum distance between a demand point and its closest facility. Constraint (2) guarantees that every demand point will be covered by at least one facility, while constraint (3) ensures that exactly p facilities are chosen. Constraint (4) guarantees that the solution value is the radius of the largest covering circle and constraint (5) refers to the binary decision variables.

The mathematical formulation above may be used to solve the problem. However, due to a massive computational explosion in the number of binary variables as represented by the number of potential circles, the implementation is not practical for larger n. For convenience, the formulation is reproduced here as it serves as an important basis in the proposed research that actually uses a set covering-based formulation.

The contributions of this study are as follows:

i) several enhancements supported by mathematical proofs are derived to strengthen and speed up a recently proposed relaxation-based algorithm for solving the continuous $p-$centre problem;

ii) large problem instances with up to 1332 nodes and 90 facilities are now capable of being solved for the first time;

iii) adaptations of the proposed algorithm to optimally solve two related location problems, namely, the $\alpha-$neighbour $p-$centre and the conditional $p-$centre problems, are also presented. These two variants play an important role for locating public service facilities.

The paper is organised as follows. A brief literature review is given in the next section. Section 3 describes the reverse relaxation algorithm given by Chen & Chen (2009), and proposes four new enhancements designed to significantly reduce the computational time. Section 4 shows the computational results found for large data sets taken from the TSP-library, namely $rat575$, $rat783$, $pr1002$ and $rl1323$. Section 5 describes the adaptations introduced to our algorithm to optimally solve both the $\alpha-$neighbour and the conditional $p-$centre problems. We also introduce a practical extension to the $\alpha-$neighbour problem where the required coverage for each demand point is not necessarily the same. Finally, we summarize our findings alongside further research suggestions.

2. A Brief Literature Review

Research into the $p-$centre problem first began by studying the simpler $1-$centre problem, that is, finding the location of one facility in a plane or network in order to minimise the maximum distance from a given set of demand points to the facility. Sylvester (1860) was the first to propose and to solve this problem. Since then, efficient algorithms have been developed to solve this problem quickly and efficiently, such as Elzinga & Hearn's (1972) geometrical-based algorithm. Several enhancements have been made to this algorithm to reduce its overall computational time further, see Elshaikh *et al* (2015) and references therein.

Cooper (1961) was the first to formulate the general location problem with $p > 1$. He later proposed four heuristics that could be used to solve the $p-$centre problem (Cooper(1964)). This included the multi-start heuristic that is now commonly employed to obtain an upper bound that can be embedded into optimal methods (or sophisticated metaheuristics). Minieka (1970) solved the $p-$centre problem on a network as a minimal set covering, which was then developed further by Daskin (1995) to create an efficient set-covering based algorithm. Elshaikh et al (2015) designed a powerful and adaptive variable neighbourhood search heuristic, while Elshaikh *et al* (2016) explored a pertubation method for the same problem with encouraging results.

Some real-life applications have also been modelled as the $p-$centre problem. For example, Pacheco & Casado (2004) presented a real health case study, where they located health resources such as geriatric and diabetic health care clinics in Spain using a scatter search algorithm. Drezner et al (2006) tackled the location of casualty collection points which happen in cases of mass casualty due to catastrophic events such as earthquakes or natural or man-made incidents. A five-objective optimisation problem with the $p-$centre included is utlised and tested on a case study based on Orange County in Califfornia. Wei et al (2006) solved the constrained continuous $p-$centre problem to locate 25 service facilities in Dublin, Ohio using a Voronoi-based algorithm. Kaveh & Nasr (2011) used a modified harmony search algorithm to solve the conditional and unconditional $p-$centre problem for locating bicycle stations in the city of Isfahan. Lu (2013) located urgent relief centres in Taiwan to serve the injured residents following a 7.3 Richter scale earthquake by devising a simulated annealing-based heuristic.

It is important to note that in the above real case studies, heuristic solutions to the $p-$centre problem are used exclusively. However, due to many advances in technology and computer power, alongside the latest developments in exact algorithms, it is now possible to find the optimal solutions to such problems.

Drezner (1984) proposed an exact method to solve the $p-$centre problem that used a subset of potential facility locations generated by an upper bound using the concept of maximal circles. Recent enhancements to this method have created a faster, more efficient algorithm that is used to optimally solve large problems for the first time, see Callaghan *et al* (2017).

A classic relaxation algorithm was originally suggested by Handler & Mirchandani (1979) to solve the discrete $p-$ centre problem. Chen & Handler (1987) proposed a relaxation-based algorithm for the continuous $p-$centre problem after observing that a) the solution to the $(p-1)-$centre problem is an upper bound for the $p-$centre problem and b) if the optimal solution for the sub-problem is feasible for the original problem, then it is optimal for the original problem. Chen & Chen (2009) proposed three interesting alternative algorithms to the classic relaxation algorithm which they referred to as the improved relaxation algorithm, the binary relaxation algorithm and the reverse relaxation algorithm. The improved relaxation algorithm updates the upper bound more frequently and adds k points to the subset at a time rather than just one at a time. The latter two algorithms form two new relaxation-based methods. Note that the reverse relaxation algorithm will be re-examined

and dramatically strengthened in this study.

An interesting related problem that we also investigate here is the $\alpha-$neighbour $p-$centre problem. This was first presented by Krumke (1995) with the aim to minimise the maximum distance between each demand point and its closest α facilities. This is a useful strategic problem that is used in the case of disruption so that coverage is still valid. Krumke (1995) provided an approximation algorithm to solve the problem on a network with an approximation factor of 4 when $\alpha \geq 2$. Khuller et al (2000) improved this by producing an algorithm that had an approximation factor of 3 for $\alpha \geq 3$, and 2 for $\alpha = 2$. Chen & Chen (2013) devised an optimal algorithm for the continuous $\alpha-$neighbour $p-$centre problem by adapting two well-known optimal algorithms, namely Minieka's (1970) algorithm used to solve the discrete and continuous $p-$centre problem, and the classic relaxation algorithm by Chen & Chen (2009). The authors found optimal solutions for one TSP-Library data set ($n = 439$) for the first time with $\alpha = 2$ and 3. Our method will be shown to be flexible and powerful enough to optimally solve for the first time much larger instances (e.g. up to $n = 1323$).

Another related problem which we also explore here is the conditional $p-$centre problem. This is sometimes referred to as the $(p, q)-$centre problem with the aim to locate p facilities on a plane or network given that q facilities already exist. This is a closely related problem faced by management when services need to be expanded. Minieka (1970) introduced conditional location problems on a network and Handler & Mirchandani (1979) formally presented the first conditional location problem. Chen & Handler (1993) solved the conditional $p-$centre problem by modifying the relaxation-based method for the continuous $p-$centre problem of Chen & Handler (1987) using two ways. The first modification disallowed demand points within a certain distance to an existing facility to be added to the subset, while the second one included the existing facilities when checking for feasibility for the full problem. Drezner (1989) proposed a binary algorithm for the conditional $p-$centre problem, and Chen & Chen (2010) developed this algorithm further by incorporating their earlier reverse relaxation algorithm (see Chen & Chen (2009)). This method will be revisited in Section 5.

3. An Enhancement-Based Algorithm

Chen & Chen (2009) proposed two new relaxation algorithms to solve the $p-$centre problem called the reverse relaxation algorithm and the binary relaxation algorithm. Although these

algorithms are proven to be efficient at solving the discrete $p-$centre problem, they have not been tested on several data sets in the continuous case. We conducted a detailed experiment using all values of k ($k = 1, \ldots, 10$) and found that the reverse relaxation algorithm performs relatively better on average. For brevity, the detailed results are not given here but can be found in Callaghan (2016). We therefore focus on this algorithm which is given in Figure 1. Note that in Step 3 of Figure 1, the sub-problem is said to be feasible if the number of facilities found to cover the subset of demand points for a given coverage distance when solving the set covering problem, is $\leq p$. For simplicity the general set covering formulation which we refer to as SCF is reproduced here based on the entire set of demand points I and the set of facilities (circles) J. All other notation is given earlier in CP_1.

$$SCF\text{: Minimise} \quad \sum_{j \in J} x_j \tag{6}$$

$$\text{subject to} \quad \sum_{j \in J} A_{i,j} x_j \geq 1 \quad \forall i \in I, \tag{7}$$

$$x_j \in \{0, 1\} \quad \forall j \in J. \tag{8}$$

The objective function (6) wishes to minimise the number of facilities to open and constraint (7) ensures that all demand points are covered by at least one facility.

The rest of this section will describe the four enhancements that we developed to create a deterministic, fast and efficient algorithm that is powerful enough to solve large continuous $p-$centre problems optimally.

3.1. A Deterministic Generator for the Initial Subset (Step 2 of Figure 1)

One potential way to enhance Chen & Chen's algorithm would be to reexamine the selection of the initial subset. One scheme is to guarantee that the demand points chosen are evenly spread out over the data's distribution besides being generated in a deterministic way. This would lead to computational times that are less sensitive to the initial subset of the demand points. This idea is translated into our first enhancement, SubE1, which aims to find the minimum number of dispersed initial demand points.

1. Set the lower bound to be 0 and specify k.
2. Choose a random subset of demand points, Sub.
3. Determine whether Sub has a feasible solution, $feasible$, with objective value less than the lower bound.
    a) If $feasible$ cannot be found, generate a new lower bound by finding the smallest radius of a covering circle created from Sub that is larger than the current lower bound and go back to Step 3.
    b) If $feasible$ can be found, continue to Step 4.
4. Determine whether $feasible$ is feasible for the full problem.
    a) If it is, halt and return $feasible$ as the final solution.
    b) Otherwise add k demand points to Sub and go back to Step 3.

Figure 1: Reverse Relaxation based on Chen & Chen (2009: pp 1649)

Let us define $d'_{i,l}$ as the Euclidean distance from demand point i to demand point l, and Sub as the subset of demand points.

SubE1: An Overview

i) Firstly, the convex hull of the demand points is found, and let CS be the set of points defining the vertices of the convex hull. In this study, we used the quickhull algorithm as proposed by Barber, Dobkin & Huhdanpaa (1996).

ii) The demand point, $\hat{i} \in I$ that yields the greatest sum distance from all $i' \in CS$ is identified by

$$\hat{i} = Arg(\underset{i \in I \setminus CS}{Max}\{\sum_{i' \in CS} d'_{i,i'}\}), \tag{9}$$

and let $Sub = \{\hat{i}\}$.

iii) The algorithm then enters the third stage where

a) All $i \in I$ are allocated to their closest facility $j \in Sub$. This is represented by the following allocation matrix Al

$$Al_{i,j} = \begin{cases} 1 \text{ if } d_{i,j} < d_{i,j^*} \ \ \forall i \in I, \ \forall j, j^* \in Sub : j \neq j^*, \\ 0 \text{ else.} \end{cases}$$

Note that $d_{i,j}$ is used instead of $d'_{i,j}$ though j is both a facility and a demand

point. This is used to retain consistency of the distance between a faciltiy and a demand point.

b) The temporary facility with the most allocated demand points, j_{Max}, is then identified as

$$j_{Max} = Arg(\underset{j \in Sub}{Max}\{\sum_{i \in I} Al_{i,j}\}). \tag{10}$$

c) The demand point that is both allocated to j_{Max} and sits the furthest away from it, say i^*, is then identified as follows:

$$i^* = Arg(\underset{i \in I}{Max}(d_{i,j_{Max}} : Al_{i,j_{Max}} = 1). \tag{11}$$

d) The demand point i^* is then added to the initial subset (e.g., $Sub = Sub \cup \{i^*\}$).

We will now discuss the method used to determine the cardinality of this initial subset.

Determining the minimum $|Sub|$

The minimum number of demand points, r, needed for the initial subset Sub to yield an initial solution (i.e. p circles) is found by Chen & Handler (1987) as follows:

$$r = \underset{r' \in \mathbb{N}; r' \geq 3}{Min} : \binom{r'}{3} + \binom{r'}{2} + r' \geq p, \tag{12}$$

where r' is the number of demand points in Sub, and $\binom{r'}{3}$ and $\binom{r'}{2}$ represent the number of circles formed from three and two critical demand points in Sub respectively. For example, if $p = 4$ we need $r' = 3$ which leads to 7 identified candidate circles (one interescting the 3 demand points, three intersecting a different pair each and three degenerate circles (radius=0) intersecting a single point each.

To save computational time, the algorithm finds $(r - 1)$ demand points using $SubE1$.

The number of circles created from this initial subset may not be large enough to guarantee an initial solution (i.e., the number of circles $< p$), as some circles can be discarded due to the geometry of the critical demand points. If this is the case, the algorithm returns back to step 4 and adds another demand point to the initial subset one by one until an initial solution is obtained. The algorithm for SubE1 can be found in Figure 2.

This method allows the distribution of the data set to be embedded into the search. This

1. Set $Sub = \emptyset$. Find the demand points $i' \in I$ that form the vertices of the convex hull, defined as the set CS.
2. Find the demand point $\hat{i} \in I \setminus CS$ that has the largest sum distance from all $i' \in CS$ using Equation (9). Set $Sub = Sub \cup \{\hat{i}\}$.
3. Determine the value of r using Equation (12).
4. While $|Sub| < r$ do:
   Allocate all $i \in I$ to their closest $j \in Sub$. Determine j_{Max} using Equation (10) and then find the furthest demand point $i^* \in I$ that is allocated to j_{Max} using Equation (11). Set $Sub = Sub \cup \{i^*\}$.
5. Find all circles made by one, two or three demand points from Sub. If the number of circles is $\geq p$, return Sub as the initial subset and stop.
   Else, set $r = r + 1$ and go back to Step 4.

Figure 2: Initial Subset Algorithm (SubE1)

means the initial subset is proportionally dispersed over clustered areas as well as evenly spread areas, yielding a more efficient initial subset. Furthermore, by checking to see if the subset can produce an initial solution each time another demand point is added, the size of the subset is minimised.

As an example, Figure 3 shows the initial subset found using SubE1 for the data set $pr439$ where $p = 50$. There are 25 demand points in this initial subset.

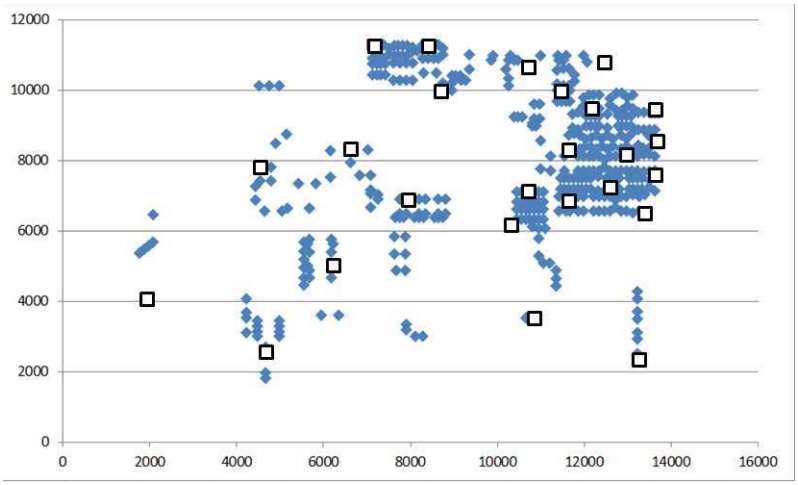


Figure 3: Initial subset for $p = 50$ using Enhancement One

Why all $i' \in CS$ are not necessarily added to the initial subset

Observation 1. *Take two points $i_1, i_3 \notin CS$ and point $i_2 \in CS$. If the angle $\angle i_1 i_2 i_3 > 90°$, then the point i_2 can be encompassed by the covering circle formed from i_1 and i_3.*

This observation is illustrated in Figure 4 where the point i_2 is encompassed by a covering circle formed from the two other points $i_1, i_3 \notin CS$.

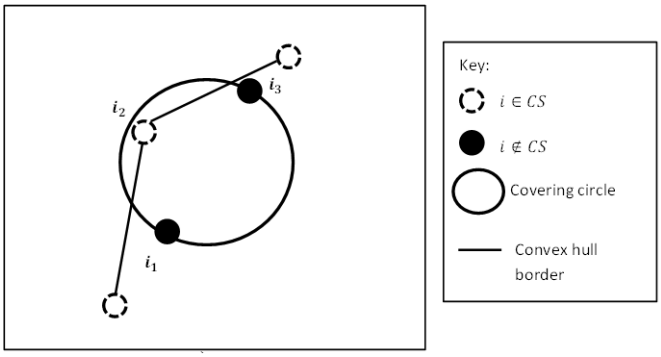


Figure 4: Observation 1

As the third part of SubE1 is designed to find a subset of dispersed points in the plane, it might seem reasonable to add all $i \in CS$ to Sub automatically. However, observation 1 demonstrates that the extreme points that form the convex hull are not necessarily critical points for their covering circles. Although SubE1 could select some points that form the vertices of the convex hull to be in Sub, it is also designed to find a good initial subset based on the data distribution and thus be more selective about the points that are chosen.

		CPU Times (secs)					
		Chen & Chen's Results	Our Chen &Chen Implementation				Our Results With SubE1
p	Z	Best† ($k=2$)	Case of ($k=2$)	Best$^{\perp}$ ($k=3$)	Worst ($k=1$)	Average^{+}	Best$^{\perp}$ ($k=2$)
10	1716.510	0.84	2.41	4.18	2.27	29.65	1.98
20	1029.715	2.63	3.02	3.22	2.31	18.94	2.12
30	739.193	6.16	7.62	3.75	5.05	27.10	42.39
40	580.005	93.38	66.54	38.69	29.29	57.96	38.96
50	468.542	207.45	95.46	63.62	69.20	127.83	67.84
60	400.195	62.19	161.96	24.49	77.55	54.59	14.18
70	357.946	103.28	461.17	57.12	52.18	107.24	45.33
80	312.500	172.59	68.12	41.62	620.60	188.09	52.39
90	280.903	157.07	49.70	63.24	274.08	85.05	85.45
100	256.680	60.40	68.13	17.32	119.36	51.92	31.59
Average		**86.60**	**98.41**	**31.73**	**125.18**	**74.81**	**38.22**

† Results reported from Chen & Chen's paper with $k = 2$.
$^{+}$ Average over $k = 1, \ldots, 10$.
$^{\perp}$ Best result for $k = 1, \ldots, 10$.

Table 1: Results comparing the Reverse Relaxation Algorithm with and without SubE1

Table 1 shows the comparison between the computational times for Chen & Chen's results, our results based on our implementation of Chen & Chen algorithm, and our results with SubE1 for the reverse relaxation algorithm applied to the TSP-Library data set $pr439$. These results may suggest that initially the first enhancement does not improve the computational time when compared with our original results. However, as stated before, SubE1 has the advantage of being deterministic. When compared with Chen & Chen's results, it can be seen that SubE1 was considerably faster on average. It is worth noting that such comparison may not be valid as the other authors used a relatively older computer, older version of cplex and above all different coding style. In brief, they used a 3.2 GHz Intel Pentium 4 computer while we ran our tests on a much faster machine, namely, HP Elitebook 8570w with 12 GB memory. For the optimisation part, we used the IBM cplex 12.6 console while they solved their models with cplex 7.5. An attempt to assess the performance of the two machines can be explored in Dongarra (1992) if necessary. To avoid any unnecessary confusion, we also reported the results of our implementation of Chen &Chen algorithm for the same case when $k = 2$ using our code, our CPLEX optimiser and our machine. These results show that the original implementation is relatively much slower as the overall average of around 98 seconds is reported compared to about 38 seconds only for SubE1. Given that the original algorithm is non-deterministic, some runs may obviously be much faster than others. As an example, our first run (when $k = 3$) happens to be the fastest on average as a better initial subset may have been randomly selected. Therefore, it is promising to see the computational time taken by SubE1 to be much closer to the best initial solution generated by the random selection, besides that the CPU time with SubE1 is less sensitive to the value of p. Above all, a deterministic method such as SubE1 is more reliable as it can be replicated by other researchers.

3.2. An Efficient Scheme for Adding Demand Points (Step 4b in Figure 1)

Chen & Chen's reverse relaxation algorithm states that the demand points added to the subset are the k furthest ones from their allocated facilities. This scheme, though interesting and simple to use, could be improved further by identifying its weakness through the following 'worst-case scenario' example.

In order to explain the second enhancement, we define $R(K)$ as the radius of the smallest circle encompassing all points in K (where $K \subset I$). The closure of circle C_j is defined as

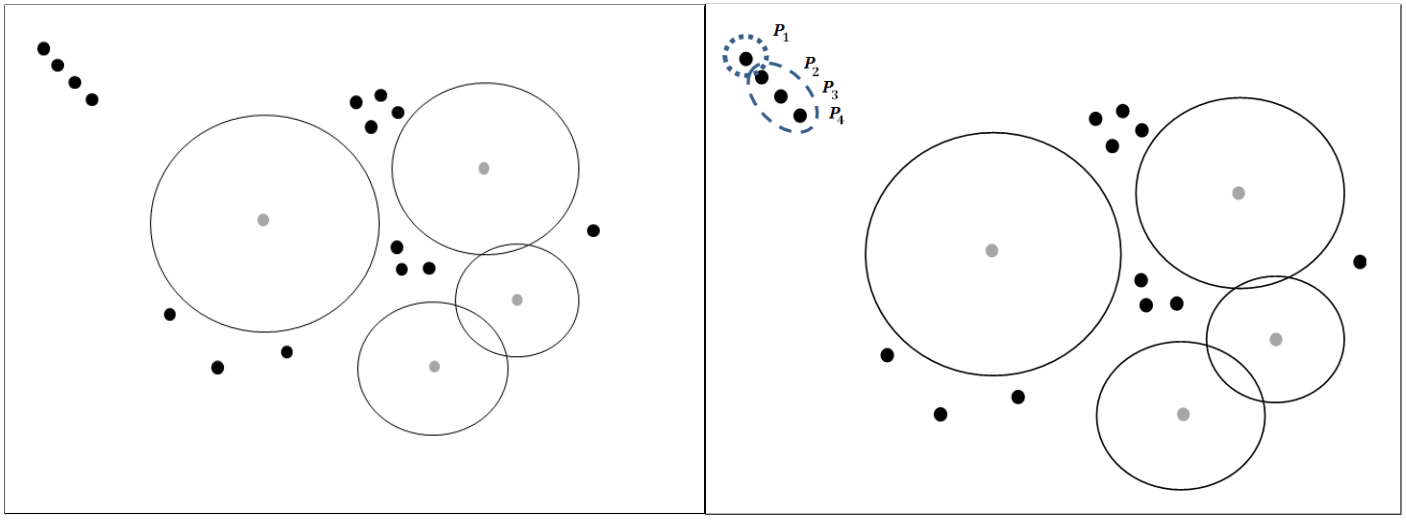
follows.

Definition 1. *The* **closure of circle** C_j *is the set of demand points covered by circle* C_j*:*

$$Cl_j = \{i \ \in I |\ d_{i,j} \leq r_j\} \ \forall \ j = 1 \ldots m.$$

An Illustrative Example

Figure 5a demonstrates a solution for the relaxed subset where $p = 4$. The grey dots are the facility locations (circle centres) for the subset's solution. The black dots are the uncovered demand points in the full problem. The subset's solution is not feasible for the full problem as there are still some uncovered demand points.



(a) (b)

Figure 5: Adding k furthest demand points example

As an example consider $k = 4$. According to the original algorithm, all uncovered demand points must now be allocated to their nearest facility and the k furthest points will be added to the subset. In this example, the first demand point that will be allocated to the subset is the one encompassed by its own dotted circle and labelled P_1 in Figure 5b. Adding P_1 to the subset could contribute to a good result as it would guide the algorithm towards the optimal solution.

However, the remaining three demand points that will be added to the subset are the three demand points circled with a dashed line and labelled P_2, P_3 and P_4 in Figure 5b. The addition of these three demand points to the subset does not contribute to improving the solution as the new information is now redundant. This is because $R(K \cup \{P_1\}) \geq$

$R(K \cup \{P_k\}), k = 2, 3, 4$, as P_1 lies further from the common closest circle C_j. Adding P_2, P_3 or P_4 to *Sub* would mean there were extra calculations that are both redundant and time consuming. This claim is stated in Lemma 2.

Lemma 1. *Take two demand points, i^1 and i^2, that are not encompassed by C_j. If $d_{i^1,j} > d_{i^2,j}$, then $R(K \cup \{i^1, i^2\}) > R(K \cup \{i^2\})$.*

Proof.

$$R(K \cup \{i^1, i^2\}) = \frac{r_j + d_{i^2,j} + (d_{i^1,j} - \nu)}{2} - \epsilon$$

where $0 \leq \nu \leq d_{i^2,j}, \epsilon$ a small positive value and

$$R(K \cup \{i^2\}) = \frac{r_j + d_{i^2,j}}{2} - \epsilon.$$

$$d_{i^1,j} > d_{i^2,j}.$$

$$\implies \frac{d_{i^1,j} - \nu}{2} > 0,$$

$$\implies \frac{r_j + d_{i^2,j} + (d_{i^1,j} - \nu)}{2} - \epsilon > \frac{r_j + d_{i^2,j}}{2} - \epsilon,$$

Therefore $R(K \cup \{i^1, i^2\}) > R(K \cup \{i^2\})$ □

Enhancement two, which will be referred to as AddE2, aims to overcome the possible shortcoming of adding redundant points by slightly altering Chen and Chen's selection rule. In order to explain AddE2, the following definition is given.

Definition 2. *An* **artificial circle** *is a candidate circle whose radius, r_j, has been enlarged to Z_t, where Z_t is the subset's solution value at iteration t.*

At any iteration t, all circles' radii are artificially increased to size Z_t and the solution is analysed again to see what demand points remain uncovered. This allows more demand points to be covered whilst not compromising the solution quality.

Figure 6 demonstrates the usefulness of artificial circles. The demand point P_5 appears to be uncovered by circle C_j. However, once the smaller circle is transformed into an artificial circle (the dashed circle) by increasing its radius r_j to size Z, it then becomes clear that point P_5 is now covered based on the solution value Z.

The enhancement AddE2, which is described in Figure 7, uses artificial circles to enhance

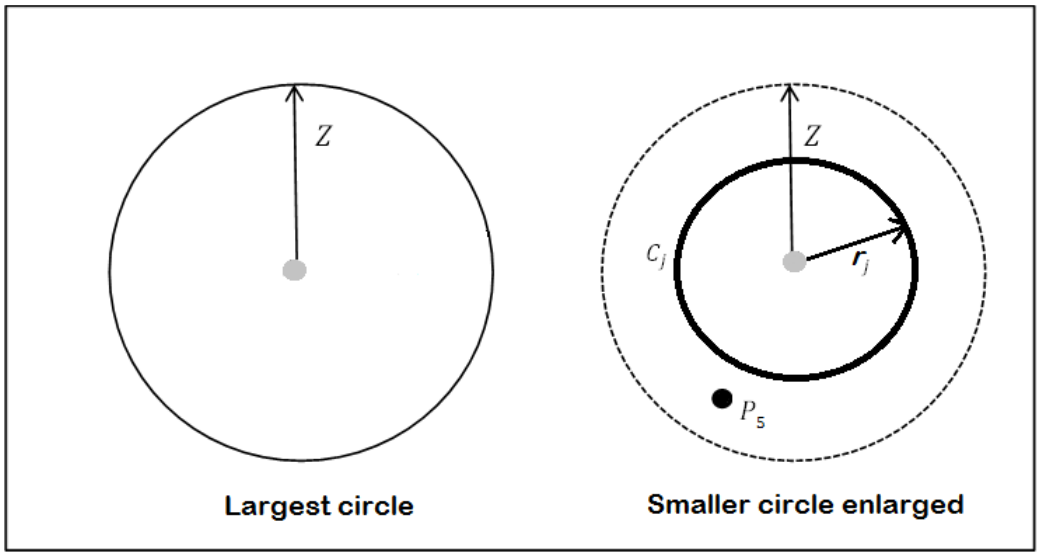


Figure 6: Construction of an artificial circle with radius Z (r_{Max})

and improve the selection technique when choosing the k demand points to be added to the subset.

1. Input: set of solution circles (facilities) F and value k.
2. Allocate all the demand points $i \in I$ to their closest facility $j \in F$ and define the allocation matrix $A_{i,j}$ where $A_{i,j} = 1$ if demand point i is allocated to facility j, else $A_{i,j} = 0$. Record $Z = Max(r_j : \ j \in F)$.
3. Create the artificial circles by artificially increasing all r_j so that $r_j = Z \ \forall \ j \in F$.
4. Update the allocation matrix such that $A_{i,j} = 1$ if demand point i is both allocated to facility j and $d_{i,j} > Z$.
5. Find the k furthest demand points from their allocated facility such that no more than one demand point is selected from each facility.

Figure 7: Point Selection Algorithm (AddE2)

AddE2 was first added on its own to the reverse relaxation algorithm and tested on the data $pr439$ where $p = 10, \ldots, 100$ and $k = 1, \ldots, 10$. This yielded an improvement with an average reduction in computational time of 10.60%. Although this is an encouraging result, fast results such as these cannot be replicated with AddE2 alone as the algorithm is not deterministic. To achieve the desired outcome, both SubE1 and AddE2 are incorporated into the reverse relaxation algorithm. The results, given in the last column of Table A.1 in the Appendix, show that the deterministic algorithm with these two enhancements yields the optimal solution in a faster time with an average decrease of computational time of 11.30%.

3.3. Jump-Based Lower Bound Update (Step 3a in Figure 1)

The reverse relaxation algorithm usually requires a large number of iterations due to the algorithm constantly updating the lower bound. However, many of these adjacent 'jumps' to the next lower bound could be redundant. The third enhancement, which will be referred to as DyJumpE3, takes this drawback into account by providing a more efficient updating scheme by redefining what the next lower bound will be set to. Instead of setting the next lower bound as the smallest radius larger than the current lower bound, it will be set to the $jump^{th}$ smallest radius where the value of $jump$ is determined dynamically.

A similar approach was first proposed for the discrete $p-$centre problem by Al-Khedhairi & Salhi (2005), where the authors suggested the use of the second lower bound rather than the first. As the algorithm may require several iterations until the optimal solution is reached by the lower bound, we set the $jump^{th}$ lowest bound to $LB_0 = LB_{jump}$ where

$$LB_{jump} = min\{d_{i,j} : d_{i,j} > LB_{jump-1} \; \forall \; i \in I, j \in J\}. \tag{13}$$

Defining the backward jump

Note that if an upper bound is found when $jump = 2$, the next lower bound to consider is the single value before that was missed as demonstrated by Al-Khedhairi & Salhi. However, if $jump > 2$, this scheme is no longer valid as there is more than one value missed.

One option is to find the next value using the binary (bisector) method. In other words, the missed value that is halfway between the current lower bound and upper bound is taken as the new bound to be evaluated (as it can be determined as either a lower bound or an upper bound). The option which we adopt is to use a strategic increase leading to the very first missed lower bound (i.e., upper bound) and then continue from this point to decrease the value systematically until a lower bound is identified meaning that the upper bound previously used is then the optimal solution guaranteeing no further checking.

One may think that setting $jump$ to a large number will avoid many redundant iterations as the lower bound is updated fewer times. However, this is not always true as when demand points are added to the relaxed subset, more circles are generated that have radius r_j such that $LB_{q(t-1)} < r_j < LB_{qt}$ at iteration t and $jump = q$. A scheme that balances the use of large jumps whilst minimising the number of circles to check through once an upper bound

is established is given next.

A dynamic jumping scheme

A dynamic scheme where $jump$ is initially set to be large, and then slowly decreased during the search is presented. This is a non-increasing but discontinuous function of iterations $jump(t)$ with t denoting the t^{th} iteration and $jump(t) \in \mathbb{N}$. As the value of $jump$ will change throughout the search, the computational time savings for $0 \leq q < jump$, $(No_q^{(jump)})$, can be defined for any value of $jump$ as follows:

$$No_q^{(jump)} = \begin{cases} \alpha - \lfloor \frac{\alpha}{jump} - 1 \rfloor & \text{if } q = 0, \\ \alpha - \lfloor \frac{\alpha - q}{jump} + 1 + (jump - q) \rfloor & \text{if } q > 0, \end{cases} \tag{14}$$

where $\alpha = q \bmod jump$.

Let $jump_{Max}$ be the maximum number of jumps, such that $2 \leq jump(t) \leq jump_{Max} \ \ \forall t$. Let N_u^t be the number of uncovered demand points at iteration t. Initially we set $jump = jump_{Max}$. The scheme uses the proportion of uncovered demand points as defined in Equation (15).

$$jump(t) = \lfloor jump_{Max} - (jump_{Max} \times (1 - \frac{N_u^t}{N}) + 0.5) \rfloor. \tag{15}$$

This scheme was tested on the data set $pr439$ with $jump_{Max} = 10$ and $k = 1, \ldots, 10$ with the best result found when $k = 4$ with an average of 6.40 seconds whereas the worst computational time recorded was for $k = 1$ which required nearly two and a half times more computational time (i.e., 16.95 seconds).

3.4. A Dynamic Scheme for the Determination of k (Step 4b in Figure 1)

When the enhanced algorithm did not include DyJumpE3, the best results were often obtained when $k = 2$ or 3. However when DyJumpE3 was included, the best computational times were found when $k = 4$. This suggests that the value of k is critical to the success of the algorithm.

One way to identify the most appropriate estimator for k would be to perform a thorough statistical analysis. However, this requires performing experiments with the expectation that a similar performance will remain valid when tested on new data sets. Another, more robust, approach is to develop a self-adaptive method that will learn as the search progresses while

incorporating the characteristics of the $p-$centre problem. Our final enhancement, PointE4, aims to implement such an idea where we aim to determine the number of demand points k during the algorithm based on the number of uncovered points. This scheme shares some similarities with the dynamic jumping scheme described previously. In other words, PointE4 seeks to find a balance between adding more demand points at the beginning, and decreasing k as the algorithm starts to slow. This is achieved by setting k to be a certain percentage of uncovered demand points so a proportionally small amount of demand points can be added to the subset at each iteration.

Previous results show that the reverse relaxation algorithm generally works best when $k > 1$. Therefore, at iteration t we set k to be

$$k(t) = Min(p, Max(2, \lfloor (0.02 \times N_u^t) + 0.5 \rfloor)). \tag{16}$$

In other words, the minimum value of k can be at least 2 while being bounded by a maximum of p.

3.5. The Enhanced Reverse Relaxation Algorithm

Figure 8 describes the Enhanced Reverse Relaxation algorithm ($ERRA$) incorporating all four enhancements.

4. Computational Results of ERRA

$ERRA$ was tested on larger instances, namely the TSP-library data sets $rat575$, $rat783$, $pr1002$ and $rl1323$. Each data set was given a time limit of 24 hours for each value of p, and the result found is either the optimal solution or the best lower bound established within the time limit. The latter can be used, if necessary, as bounds when evaluating heuristic solutions in general.

Tables 2-6 are structured in the following way. The first column, titled p, shows the number of required facilities. The second column, titled Z_H, gives the upper bound found from the best heuristic solution obtained in Elshaikh *et al* (2015). This value was used as an upper bound on the radius whenever updating the set of candidate circles was required. The third column gives the optimal solution, Z^*, or the best lower bound shown by $^\perp$ if no

1. Set the lower bound $LB = 0$, $c = 1$, $t = 0$ and $jump_{Max} = 10$.
2. Select the initial subset of demand points using Initial Subset Algorithm given in Figure 2.
3. Set $t = t + 1$. Determine if Sub has a feasible solution, $feasible$, with a solution value, $Z_t \leq LB$.
   a) If $feasible$ cannot be found, determine the value of $jump$ using Equation (15) and find the $jump^{th}$ smallest lower bound, LB_{jump}, that is larger than LB. Set $LB = LB_{jump}$ and go back to Step 3.
   b) Otherwise if $feasible$ is found, determine N_u^t, and continue to Step 4.
4. Determine whether $feasible$ is feasible for the full problem.
   a) If it is, determine whether Sub has a feasible solution, $feasible$, with a solution value $\leq LB_{(jump-c)}$.
      If not, return $LB_{jump-(c-1)}$ as the final solution.
      Else set $c = c + 1$ and repeat Step 4(a).
   b) Determine the value of k using Equation (16). Add k demand points to Sub using the Point Selection Algorithm in Figure 7 and go to Step 3.

Figure 8: The Enhanced Reverse Relaxation Algorithm ($ERRA$)

optimal solution is obtained. The final three columns state the total number of iterations of $ERRA$, the total number of times the lower bound was updated and the total computational time in seconds (if less than 24 hours), respectively.

Results show that optimal solutions were found in a reasonable computational time for the larger data sets with some results being discovered for the first time, such as those for the data set $rl1323$ when $p < 30$. This suggests that $ERRA$ is particularly effective for smaller values of p.

Furthermore, if we compare the best upper bound solution found (Z_H) to the best lower bound found within the time limit of 86400 seconds, we can observe that the lower bound is very close to the value of the best upper bound, with the worst-case being a 4.4% difference ($n = 783, p = 80$). This confirms that the solutions found by the VNS-based metaheuristic are either optimal or near optimal for those instances with unconfirmed optimal solutions. On the other hand, referring to Table 2, we see that the VNS solutions can be quite far from the optimal ones.

	Best Heuristic	Enhanced Reverse Relaxation			
p	Z_H	Z^*	# Iterations	# LB Updates	CPU Time (secs)
10	1716.510	1716.510	40	11	0.78
20	1169.540	1029.715	53	9	1.48
30	975.000	739.193	68	9	2.86
40	874.271	580.005	93	13	9.97
50	580.005	468.542	107	26	13.79
60	570.088	400.195	99	12	10.00
70	503.271	357.946	130	25	16.77
80	467.039	312.500	126	19	14.95
90	391.511	280.903	139	17	20.83
100	315.486	256.680	128	7	15.55
Average	**756.272**	**614.218**	**98**	**15**	**10.70**

Z_H: Heuristic solution found by Elshaikh *et al* (2015)

Table 2: Results for TSP-Lib *pr*439 using *ERRA*

	Best Heuristic	Enhanced Reverse Relaxation			
p	Z_H	Z^*	# Iterations	# LB Updates	CPU Time (secs)
10	67.926	67.926	110	68	11.42
20	45.621	45.475	342	233	590.42
30	35.556	35.556	335	188	1412.07
40	30.265	30.063	434	253	76647.60
50	26.173	25.826	332	159	16354.00
60	23.662	23.163	294	117	22277.80
70	21.059	20.858	283	100	28210.70
80	19.510	19.026	253	68	9789.67
90	17.923	17.460	239	60	1114.37
100	16.511	16.420	243	53	3696.07
Average	**30.421**	**30.178**	**287**	**130**	**16010.41**

Z_H: Heuristic solution found by Elshaikh *et al* (2015)

Table 3: Results for TSP-Lib *rat*575 using *ERRA*

	Best Heuristic	Enhanced Reverse Relaxation			
p	Z_H	Z^*	# Iterations	# LB Updates	CPU Time (secs)
10	79.313	79.313	151	102	24.14
20	53.441	53.332	386	268	1181.29
30	42.395	42.307	638	460	39558.80
40	35.962	$35.249^{\perp}$	417	259	86400.00
50	31.184	$30.647^{\perp}$	401	236	86400.00
60	28.053	$27.067^{\perp}$	276	134	86400.00
70	25.446	$24.521^{\perp}$	250	114	86400.00
80	23.560	$22.519^{\perp}$	261	112	86400.00
90	21.710	$20.940^{\perp}$	265	103	86400.00
100	20.334	$19.526^{\perp}$	254	90	86400.00
Average	**36.140**	$---$	**330**	**188**	**64556.42**

Z_H: Heuristic solution found by Elshaikh *et al* (2015)

$^{\perp}$ Best lower bound found within 86400 seconds.

Table 4: Results for TSP-Lib *rat*783 using *ERRA*

	Best Heuristic	Enhanced Reverse Relaxation			
p	Z_H	Z^*	# Iterations	# LB Updates	CPU Time (secs)
10	2389.36	2389.36	119	69	17.97
20	1609.54	1607.53	471	336	5066.19
30	1231.36	1231.36	330	185	2072.17
40	1030.40	1021.41	307	159	1091.56
50	901.455	895.342	333	170	10874.10
60	801.474	795.709	328	137	23724.40
70	727.154	725.431	366	142	10545.50
80	664.798	655.746	269	89	1492.98
90	604.152	604.152	270	67	670.39
100	559.017	555.662	256	51	391.07
Average	**1051.870**	**1048.170**	**305**	**141**	**5594.63**

Z_H: Heuristic solution found by Elshaikh *et al* (2015)

Table 5: Results for TSP-Lib $pr1002$ using $ERRA$

	Best Heuristic	Enhanced Reverse Relaxation			
p	Z_H	Z^*	# Iterations	# LB Updates	CPU Time (secs)
10	2897.49	2987.49	328	242	666.71
20	1886.82	1868.92	674	543	5485.95
30	1466.97	1466.97	858	670	81009.20
40	1236.38	$1225.74^{\perp}$	666	486	86400.00
50	1060.82	$1051.82^{\perp}$	557	384	86400.00
60	941.870	$930.977^{\perp}$	472	297	86400.00
70	844.967	$841.578^{\perp}$	571	323	86400.00
80	774.764	$770.532^{\perp}$	495	267	86400.00
90	720.625	706.145	428	202	9863.38
100	662.936	$658.267^{\perp}$	424	195	86400.00
Average	**1249.364**	$---$	**547**	**361**	**61542.52**

Z_H: Heuristic solution found by Elshaikh *et al* (2015)
$^{\perp}$ Best lower bound found within 86400 seconds.

Table 6: Results for TSP-Lib $rl1323$ using $ERRA$

5. Adapting ERRA to two related location problems

In this section, we show the effectiveness and flexibility of ERRA to easily adapt to related p-centre problems that have a considerable significance in practice. These include the α -neighbour p -centre and the conditional $p-$ centre problems.

5.1. The $\alpha-$Neighbour $p-$Centre Problem

This problem aims to minimise the maximum distance between each demand point and its $\alpha-$ closest facility (where $\alpha < p$). For example, if $\alpha = 3$, we wish to minimise the maximum distance to the third closest facilitiy. This is equivalent to ensuring that each

demand point is covered by at least $\alpha = 3$ covering circles. In other words, Z^* is the minmax closest distance to the nearest facility that ensures all demand points are covered by at least α facilities within this maximum distance.

A strength of this classification type is that it allows for either failure or closure of $\alpha - 1$ facilities whilst ensuring that each demand point is still covered. This provides extra safety and security, which is particularly important when locating emergency facilities.

As previously stated, Chen & Chen [7] were the first to propose an optimal algorithm for the $\alpha-$neighbour $p-$centre problem by adjusting the classic relaxation algorithm. $ERRA$ shares many similar properties to the classic relaxation algorithm, and hence some adaptations for $ERRA$ are similar, though some are different. The modifications include (i) changes to the formulation, (ii) adjusting the enhancement AddE2, (iii) clarifying how the solution value is obtained and (iv) discussing the concept of co-location of facilities.

5.1.1. The Modifications of ERRA for the $\alpha-$Neighbour $p-$Centre Problem

(i) Some changes to the set-covering formulation

$ERRA$ uses a similar set covering formulation as the one given in (SCF) to solve the $p-$centre problem except that constraint (7) is replaced by constraint (18) which ensures that all demand points are now covered by at least α facilities. We refer to this new model as For^{α}_{sc}.

For^{α}_{sc}: Minimise
$$\sum_{j \in J} x_j \tag{17}$$

subject to
$$\sum_{j \in J} A_{i,j} x_j \geq \alpha \quad \forall i \in I, \tag{18}$$
$$x_j \in \{0, 1\} \quad \forall j \in J. \tag{19}$$

(ii) Adjusting the enhancement AddE2

In both the classic relaxation algorithm and $ERRA$, if the solution to the subset is not feasible for the full problem, then k demand points are added to the subset of demand points. In the case of the $\alpha-$ neighbour $p-$centre problem, Chen & Chen adjusted the classic relaxation algorithm so that the k demand points that lie the furthest from their α^{th}

nearest facility were added to the subset of demand points instead. We can therefore amend our selection of the k demand points in the following way:

1. Allocate all $i \in I$ to their α^{th} closest facility. This forms at most p clusters.

2. Find the k demand points that lie the furthest from their α^{th} closest facility such that only one demand point is selected from each cluster.

(iii) Finding the Solution Value

For the $\alpha-$ neighbour $p-$centre, the worst case scenario would be the failure of the closest $(\alpha - 1)$ facilities serving a demand point. Therefore, the solution value for this problem is also the radius of the largest covering circle.

(iv) Co-Location of Facilities

For the classic $p-$centre problem, the co-location of facilities (i.e., more than one facility situated at the same location) may not be worth it, as the extra facility could be better located within the maximum circle. However, for the $\alpha-$neighbour $p-$centre problem, this observation is no longer valid and hence co-location may be beneficial as demonstrated by the following lemma.

Let $\hat{d}_i^{\alpha}$ be the distance from demand point $i \in I$ to its α^{th} closest facility, and i' the demand point that yields the largest $\hat{d}_i^{\alpha}$ (i.e., $i' = Arg[\underset{i \in I}{Max}\ \{\hat{d}_i^{\alpha}\}]$).

Also let J' be the set of α closest circles to demand point i', and $Z^{I'}$ the corresponding optimal solution value for the $\alpha-$neighbour $p-$centre problem for a subset of demand points $I' \subset I$.

Lemma 2. *If* $\exists\ d_{i',j''} > Z^{I \setminus \{i'\}}$ *for* $j'' \in J'$*, then the co-location of facilities will improve the solution value* Z^I*.*

Proof. The optimal solution, with value $Z^{I \setminus \{i'\}}$, for the $\alpha-$neighbour $p-$centre problem covers demand points $I \setminus \{i'\}$. Therefore, Z^I must cover demand point i' with α circles. As there is at least one $j'' \in J'$ with a distance

$$d_{i',j''} > Z^{I \setminus \{i'\}}$$

to demand point i', this implies

$$Z^I > Z^{I\setminus\{i'\}}.$$

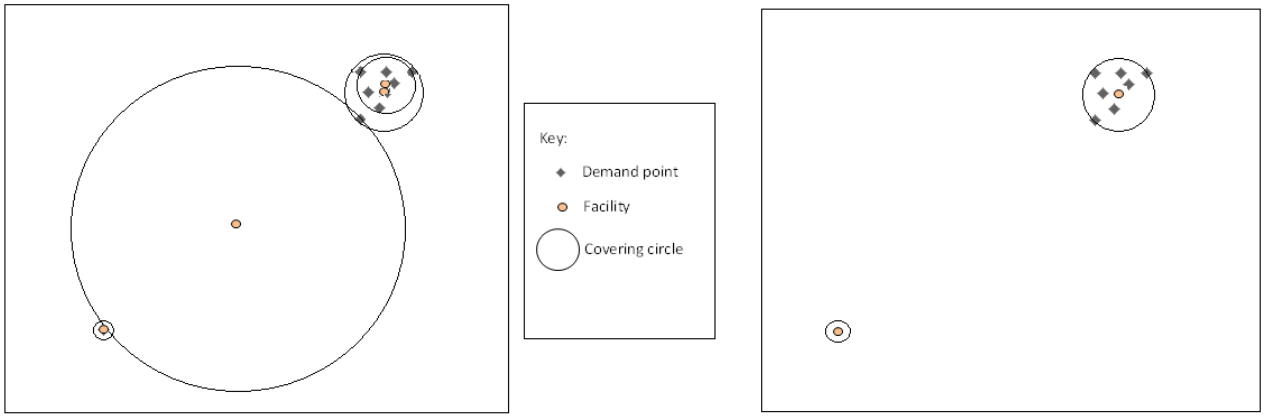
Therefore, to reduce Z^I (i.e. $Z^I = Z^{I\setminus\{i'\}}$), $p^\perp$ facilities must be co-located at a facility site $j'' \in J'$ such that $d_{i',j''} \leq Z^{I\setminus\{i'\}}$ where $1 < p^\perp \leq \alpha$. □

In order to allow co-location of facilities, constraint (19) in For^{α}_{sc} is altered to constraint (20) where x_j changes from a binary variable to an integer variable.

$$x_j \in \{0, \ldots, \alpha\} \quad \forall j \in J. \tag{20}$$

A Simple Illustrative Example

Figure 9a shows the solution to the 2−neighbour 4−centre problem for this data set where co-location is not permitted, yielding a solution value of 6.72 (i.e., maximum distance to the 2^{nd} closest facility). Figure 9b displays the solution for the 2−neighbour 4−centre problem for the same data set where co-location is allowed. In this case, two facilities are located at each site yielding a massive improvement with the new solution value of 2.83. In this particular example, the demand point i' is the one covered by the null circle and the largest circle in Figure 9a but served by two null circles in Figure 9b.



(a) Co-location is not allowed (b) Co-location is allowed

Figure 9: Solving the 2−neighbour 4−centre problem

5.1.2. Computational Results

The four adaptations were incorporated into $ERRA$ to form the adapted $ERRA$ ($AERRA$). This modified exact algorithm was used to optimally solve the $\alpha-$neighbour $p-$centre for the TSP-Library data set $pr439$ where $p = 10, 20, \ldots, 100$ and $\alpha = 2$. For comparison purposes, the results in Table 7 are also given alongside Chen & Chen's [7] results where the adapted classic relaxation algorithm was used.

		Classic Relaxation (Chen & Chen)			Adapted Enhanced Reverse Relaxation		
p	Z^*	CPU (secs)	Sub_{Max}	# Iterations	CPU (secs)	Sub_{Max}	# Iterations
10	2752.639	0.37	52	82	0.37	35	24
20	1716.510	3.04	108	162	0.84	60	37
30	1271.830	18.32	158	237	0.94	75	35
40	1008.170	27.20	196	314	3.40	120	59
50	874.271	605.65	250	389	4.53	125	63
60	739.193	978.71	260	404	7.90	154	78
70	621.742	1888.61	306	493	25.81	193	104
80	580.005	1576.88	322	515	24.80	210	112
90	530.477	1737.54	341	565	49.18	247	140
100	463.175	1443.72	352	587	26.89	234	115
Average	**1055.801**	**828.004**	**235**	**375**	**14.52**	**145**	**77**

Table 7: Optimal results for the 2−neighbour $p-$centre problem for $pr439$ using Chen and Chen 's algorithm and AERRA

Table 7's first column states the number of facilities, p, that need to be located. The second column, titled Z^*, shows the optimal solution found for the corresponding p value (i.e., the maximum distance to the 2^{nd} closest facility). The next three columns give Chen & Chen's results, with the third one referring to the total amount of computational time spent (in seconds), the fourth to the maximal size of the relaxation sub-problem (Sub_{Max}) and the fifth to the number of sub-problems (iterations) solved. The remaining three columns are ordered identically, with the results corresponding to those obtained by $AERRA$.

These results show that for the data set $pr439$, $AERRA$ was considerably better than the adapted classic relaxation algorithm used by Chen & Chen, as it required 98.25% less computational time, 38.30% fewer demand points in the subset and 79.47% fewer iterations to solve the problem optimally. These encouraging results suggest that $ERRA$ is both reliable and efficient when adapted to optimally solve the $\alpha-$neighbour $p-$centre problem.

Full results for the TSP-Library data sets $pr439$, $rat575$, $rat783$, $pr1002$ and $rl1323$ where $\alpha = 2$ & 3 can be found for the first time in Tables A.2-A.6 in the Appendix.

5.1.3. An Observation & Sensitivity Analysis

Let Z_α denote the solution value to the $\alpha-$neighbour $p-$centre problem. We observe that $Z_\alpha \geq Z_{\alpha-1}$, as the coverage need for each demand point has increased. In other words, if at least one demand point is not covered by α facilities, then at least one covering circle will need to increase in size in order to establish a feasible solution. Figure 10 demonstrates this observation for the data set $pr439$ where $p = 10, \ldots, 50$ and $\alpha = 1, 2$ and 3.

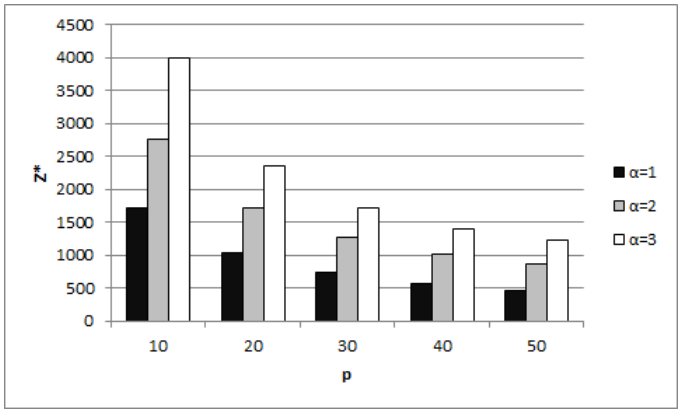


Figure 10: The optimal solution value, Z^*, for the data set $pr439$ where $p = 10, \ldots, 50$ and $\alpha = 1, 2$ & 3.

We can therefore conclude that $Z_{\alpha-1}$ can be used as an initial lower bound when solving the $\alpha-$neighbour $p-$centre problem. Furthermore, Table 8 shows the percentage increase in the solution value, and therefore additional coverage required, as the value of α increases. In this particular instance, there is a dramatic increase in the solution value on average when $\alpha = 3$ compared to $\alpha = 1$. From a managerial perspective, this means it may be more desirable to find a balance between reducing the value of α as much as possible whilst ensuring that all demand points have the required coverage. This compromise introduces an alternative setup to the $\alpha-$neighbour $p-$centre problem which will be discussed next.

5.1.4. A Variable $\alpha-$Neighbour $p-$Centre Problem

It may be beneficial from a managerial view to investigate the case where the number of times a demand point needs to be covered by a facility varies. In other words, not every demand point needs to be covered by exactly α facilities. For example, an analogy was attempted by Toregas (1971) for the set covering problem where coverage demand was not

	α=1	$\alpha = 2$		$\alpha = 3$	
p	Z^*	Z^*	% Increase	Z^*	% Increase
10	1716.51	2752.639	58.99	3989.302	132.41
20	1029.715	1716.510	66.70	2347.505	127.98
30	739.193	1271.830	72.06	1716.51	132.21
40	580.005	1008.17	73.82	1407.624	142.69
50	468.542	874.271	86.59	1226.02	161.67
Average	**906.793**	**1574.684**	**71.632**	**2137.392**	**139.39**

Table 8: Sensitivity analysis when solving the $\alpha-$neighbour $p-$centre problem where $n = 439$ and $\alpha = 1, 2$ & 3

held constant at each demand point.

$AERRA$ was modified to accommodate α_i at each demand point i, which we shall refer to as the variable $AERRA$ ($V-AERRA$). For clarity, the full formulation for the $V-AERRA$, $For_{sc}^{\alpha_{alt}}$, is given here.

$$For_{sc}^{\alpha_{alt}}\text{: Minimise} \quad \sum_{j\in J} x_j \tag{21}$$

$$\text{subject to} \quad \sum_{j=0}^{m} A_{i,j}x_j \geq \alpha_i \quad \forall i \in I, \tag{22}$$

$$x_j \in \{0, \ldots, \alpha_{Max}\} \quad \forall j \in J, \tag{23}$$

where

α_i is the required number of times demand point i needs to be covered, and α_{Max} is a maximum preassigned threshold. All other definitions are as previously given.

The objective function (21) wishes to minimise the number of facilities. Constraint (22) ensures that every demand point i is covered by at least α_i facilities, and constraint (23) represents the integer decision variable x_j that allows co-location up to α_{Max}.

For simplicity, and to provide the possibility for other researchers to conduct similar experiments, in our investigation we allocated the α_i value for each demand point based on its position in the data set. For each demand point $i \in I$ we set

$$\alpha_i = \begin{cases} 1 & \text{if } i \bmod 3 = 1, \\ 2 & \text{if } i \bmod 3 = 2, \\ 3 & \text{if } i \bmod 3 = 0. \end{cases} \tag{24}$$

This therefore demonstrates a simple example where the demand points are assigned one of three potential α values. This scenario could be adapted when appointing an α value based on the data's distribution areas (i.e. clustered, semi-clustered and sparse) or the demand point's importance ranking (i.e. high, medium or low).

Computational Results

The $V - AERRA$ was tested on the TSP-Library data sets $pr439$, $rat575$, $rat783$, $pr1002$ and $rl1323$. The results are given in Tables 9-13, which are structured as follows. The first column shows the number of facilities located, p, while the second column displays the optimal solution value, Z^*, or the best lower bound obtained shown by $\perp$ if no optimal solution is found. The last three columns show the total computational time in seconds, the maximum number of demand points in the subset, Sub_{Max}, and the total number of iterations required respectively.

p	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	3855.779	1.41	49	67
20	2298.267	1.87	68	94
30	1636.784	2.03	87	94
40	1381.179	4.75	110	142
50	1125.972	3.40	115	81
60	936.833	8.85	148	154
70	850.827	13.91	151	245
80	748.853	11.88	168	177
90	637.377	20.72	205	170
100	586.090	15.70	198	154
Average	**1405.796**	**8.45**	**129**	**137**

Table 9: Results for the variable $\alpha-$neighbour $p-$centre problem for $pr439$

p	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	131.787	6.06	45	67
20	88.996	18.30	92	88
30	64.979	79.24	129	112
40	55.322	341.73	178	179
50	46.507	531.97	209	165
60	41.599	914.78	246	187
70	38.108	779.68	225	255
80	34.567	653.07	260	190
90	31.847	16573.05	345	257
100	29.904	2200.61	302	231
Average	**56.362**	**2209.90**	**206**	**170**

Table 10: Results for the variable $\alpha-$neighbour $p-$centre problem for $rat575$

p	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	157.728	21.01	59	83
20	105.643	26.03	94	82
30	76.820	199.37	141	177
40	65.526	1423.54	200	260
50	55.626	5404.38	267	261
60	49.323	33549.65	296	342
70	$44.769^{\perp}$	86400.00	309	268
80	40.812	11710.84	366	253
90	$37.962^{\perp}$	86400.00	373	252
100	35.471	34130.75	440	265
Average	$---$	**25926.56**	**255**	**224**

$^{\perp}$ Best lower bound found in 86400 seconds

Table 11: Results for the variable $\alpha-$neighbour $p-$centre problem for $rat783$

p	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	5250.952	2.23	43	50
20	2962.790	86.57	106	116
30	2324.059	48.27	127	83
40	1948.236	1010.83	200	190
50	1676.313	6813.59	257	236
60	1494.094	15979.66	320	285
70	1367.708	15070.05	325	277
80	1226.020	5784.52	356	201
90	1155.231	7296.74	363	237
100	1068.878	2628.73	352	189
Average	**2047.428**	**5472.12**	**244**	**186**

Table 12: Results for the variable $\alpha-$neighbour $p-$centre problem for $pr1002$

p	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	6097.429	0.98	37	21
20	3677.728	345.055	132	131
30	2789.975	9415.31	209	373
40	2313.391	31059.45	280	378
50	2022.358	23532.27	299	379
60	1776.000	31290.30	348	432
70	$1604.88^{\perp}$	86400.00	303	418
80	$1464.19^{\perp}$	86400.00	318	310
90	$1351.72^{\perp}$	86400.00	309	243
100	$1264.73^{\perp}$	86400.00	370	295
Average	$- - -$	**44124.34**	**261**	**298**

$^{\perp}$ Best lower bound found in 86400 seconds

Table 13: Results for the variable $\alpha-$neighbour $p-$centre problem for $rl1323$

Optimal solutions were found using $V - AERRA$ in a reasonable amount of computational time, including the largest data set $rl1323$. This is a useful result, especially from a managerial perspective, as alternating the number of facilities to cover each demand point allows more flexibility to study different options and tradeoffs in practice.

5.2. The Conditional p−Centre Problem

The $(p, q)-$centre, or conditional $p-$centre, problem wishes to locate p facilities given that q facilities already exist. Drezner (1989) proposed an optimal algorithm to solve this problem. First, the set of demand points are allocated to their nearest facility in Q, where Q is the set of existing facilities. The demand points, and their corresponding nearest distances to the existing facilities, are then sorted into descending order of Euclidean distance. This yields an ordered set of demand points $OrdI$ and ordered distance vector $OrdD$. A bisection algorithm is then used to solve a succession of sub-problems until the solution value for the subset is greater than or equal to the next furthest distance in the list. Chen & Chen (2010) incorporated their reverse relaxation algorithm (see Chen & Chen 2009) into Drezner's algorithm to optimally solve the sub-problems. They began by creating $OrdI$ and $OrdD$ as originally suggested. The subset of demand points was then solved using the reverse relaxation algorithm, where the subset initially consisted of the first demand point only in $OrdI$. If the solution value for the subset exceeded the next furthest distance in $OrdD$, then all the demand points are covered by this solution value and so the optimal solution has been found. Otherwise, the next demand point in the ordered list was added to the subset and the process was repeated. Chen & Chen's algorithm is described in Figure 11 and is

referred to as $CON\ CCA$.

1. Input: The set of existing facilities Q.
2. Set the lower bound, LB=0, the subset of demand points $Sub = \emptyset$ and $k = 1$.
3. Allocate all $i \in I$ to their nearest facilities $\beta \in Q$. Sort the demand points in descending order of Euclidean distance. Let $OrdD = \{M_1, M_2, \ldots, M_n\}$ be the set of ordered distances, and $OrdI = \{i'_1, i'_2, \ldots, i'_n\}$ be the set of ordered demand points.
4. Add i'_k to Sub.
5. Determine if there is a feasible solution for the $p-$centre problem for Sub with a solution value $Z^k \leq LB$.
6. If so, determine whether $Z^k \geq M_{k+1}$.
   - i) If so, determine if $Z^k < M_k$.
     - If so, return Z^k as the optimal solution value and stop.
     - Else, return $Max(Z^{k-1}, M_k)$ as the optimal solution value and stop.
   - ii) Else, set $k = k + 1$ and go back to Step 4.

   Else, set LB as the smallest radius larger than the current LB and go to Step 5.

Figure 11: Chen & Chen's (2010) algorithm for the conditional $p-$centre problem ($CON\ CCA$)

Note that the optimal solution value is determined in Step 5. It is important to observe that once $Z^k \geq M_{k+1}$, either Z^k, Z^{k-1} or M_k can be determined as the optimal solution value.

5.2.1. Enhancing Chen & Chen's Algorithm

Enhancement one: Incorporating ERRA

For small problems, the $CON\ CCA$ is very efficient. However, as the problem size increases, so does $|Sub|$. This means large subsets are required to be solved optimally, which leads to computational issues such as excess memory. We therefore incorporated $ERRA$ into the $CON\ CCA$ with an aim to create a more efficient algorithm that can solve larger problems.

Enhancement two: An Efficient Initial Sub

The initial Sub in the $CON\ CCA$ consists of a single demand point. However, this can be improved by adding a further $(p-1)$ demand points, as there must be at least this number of demand points in Sub to find a meaningful initial feasible solution for the $p-$centre problem. We therefore set the initial $Sub = \{i'_1, i'_2, \ldots, i'_p\}$ where $i' \in OrdI$.

Enhancement Three: Adding more than one demand point to Sub

Each time feasibility cannot be found for the full problem, $CON\ CCA$ adds one demand point only to Sub and repeats the process again. This approach is effective on smaller data sets, but as the problem size increases, so does the final cardinality of Sub. Therefore, the process of adding one demand point at a time yields many iterations and hence greatly increases the total amount of computational time. Thus, instead of adding the next demand point only in the ordered list to Sub, the next k' demand points are added where $k' > 1$. For simplicity, we set $k' = 2$ in our experiments though other values could be explored.

For clarity, the enhanced algorithm, $CON\ ERRA$, that incorporates $ERRA$ and the two enhancements given above, is provided in Figure 12.

1. Input: The set of existing facilities Q.
2. Set the subset of demand points $Sub = \emptyset$, and logical variable tracker $TAG = True$.
3. Allocate all $i \in I$ to their nearest $\beta \in Q$. Sort the demand points in descending order of Euclidean distance. Let $OrdD = \{M_1, M_2, \ldots, M_n\}$ be the set of ordered distances, and $OrdI = \{i'_1, i'_2, \ldots, i'_n\}$ be the set of ordered demand points.
4. Set $Sub = \{i'_1, \ldots, i'_p\}$ and $k = p$.
5. Solve the $p-$centre problem for Sub using $ERRA$ (see Figure 8). This yields solution value Z^k.
6. Updating/termination step:

   If $TAG = True$ then (updating step)

   If $Z^k \geq M_{k+1}$

   If $Z^k < M_k$, set Z^k as *optimal*, else $Max(Z^{k-1}, M_k)$ is *optimal*.
   Set $Sub = Sub \setminus \{i'_k\}$, $TAG = False$, $k = k - 1$ and go to Step 5.

   Else, set $k = k + 2$, $Sub = Sub \cup \{i'_{k-1}, i'_k\}$ and go to Step 5.

   Else (i.e., $TAG = False$) then (termination step)

   If $Z^k \geq M_{k+1}$

   If $Z^k < M_k$, return Z^k as the optimal solution value and stop.
   Else, return $Max(Z^{k-1}, M_k)$ as the optimal solution value and stop.

   Else return *optimal* as the optimal solution value and stop.

Figure 12: The $CON\ ERRA$

5.2.2. Computational Results

The $(p, q)-$centre problem was solved using both the $CON\ CCA$ and the $CON\ ERRA$ for the TSP-Library data set $pr439$. Table 14 shows the results for both algorithms where

$q = 10$ & 20. The first column shows the number of new facilities located, where the maximum total number of facilities was 100 (i.e. $p + q \leq 100$). The next three columns correspond to the results using $q = 10$. The second column displays the corresponding optimal solution, Z^*, and columns three and four represent the total computational time in seconds for the *CON CCA* and the *CON ERRA*, respectively. The remaining three columns display the same information, but referring to the results where $q = 20$.

	$q = 10$			$q = 20$		
p	Z^*	CPU (secs)		Z^*	CPU (secs)	
		CON CCA	*CON ERRA*		*CON CCA*	*CON ERRA*
10	1429.434	1.83	11.41	981.150	0.79	2.10
20	958.188	2156.90	64.28	880.696	4.67	4.29
30	655.016	4231.22	138.42	705.780	212.90	40.23
40	558.038	9475.67	167.67	545.722	1499.01	91.78
50	439.638	3620.33	229.09	445.939	1904.46	171.03
60	394.305	3248.75	358.61	370.928	1083.08	274.27
70	356.000	2502.21	524.31	334.448	950.40	419.15
80	307.459	996.35	610.03	297.321	464.02	529.33
90	276.699	478.62	734.54	N/A	N/A	N/A
Average	**597.197**	**2967.99**	**354.80**	**570.256**	**764.92**	**191.52**

Table 14: Results for the conditional $p-$centre problem for $pr439$ where $q = 10$ & 20

Table 14 shows that *CON ERRA* performs better than *CON CCA*. On average, optimal solutions were obtained using approximately 81.5% less computational time. The efficiency of the *CON ERRA* is then demonstrated by solving the $(p, q)-$centre problem optimally for the TSP-Library data sets $rat575$, $rat783$, $pr1002$ and $rl1323$ where $q = 20$. The results are produced in Table 15. The first column represents the number of new facilities added, p, and the second and third columns show the optimal solution value, Z^*, and corresponding computational time for *CON ERRA* in seconds, respectively. Table 15 shows that, with the exception of $rl1323$ and $p = 70$ where memory was exceeded, optimal solutions were obtained for all data sets in a reasonable amount of time. In addition, it is encouraging to note that all solutions were obtained within a maximum computational time of 23159.05 seconds with those instances for $p \leq 30$ requiring less than 5 minutes.

	$rat575$		$rat783$		$pr1002$		$rl1323$	
p	Z^*	CPU (secs)	Z^*	CPU (secs)	Z^*	CPU (secs)	Z^*	CPU (secs)
10	45.113	1.59	52.660	2.45	1595.712	1.34	1860.859	2.15
20	39.665	16.62	47.184	18.42	1362.369	40.06	1644.258	35.56
30	32.104	97.47	37.165	182.57	1147.007	264.17	1352.777	603.99
40	26.673	287.22	32.466	1210.18	957.209	1233.67	1111.14	2995.96
50	23.345	627.22	28.306	3010.33	845.932	3561.78	973.382	7971.74
60	21.412	1114.18	25.775	6441.97	755.811	4526.57	857.13	9866.43
70	19.559	1903.97	23.505	13975.64	682.825	6454.95	1137.754^	2391.66
80	17.901	2131.41	21.552	14764.33	622.268	6460.20	718.441	23159.05
Average	**28.222**	**772.458**	**33.577**	**4950.736**	**996.142**	**2817.84**	− − −	**5878.32**

^ Best upper bound obtained (with corresponding time) due to memory issues

Table 15: Results for the conditional $p-$centre problem where $q = 20$

6. Conclusions and Suggestions

This paper investigates and enhances a relatively new relaxation-based algorithm proposed by Chen & Chen (2009) to optimally solve the continuous $p-$centre problem. Effective neighbourhood reductions are introduced to make the enhanced algorithm both deterministic and more efficient. This is empirically confirmed on five TSP-Library data sets where $n \leq 1323$. Overall, the results show that the enhanced algorithm is faster than its original counterpart. Using the enhanced algorithm, optimal solutions for larger instances, in particular, $n = 1323$ and $p = 10, 20$ & 30 were also obtained for the first time. The enhanced algorithm was also adapted to solve the $\alpha-$neighbour $p-$centre problem, with parameter α either a constant or allowed to vary over the demand points, as well as the conditional problem where a given number of facilities already exist. Optimal solutions were obtained for these two related location problems using the same TSP-Library data sets.

In this study, the initial lower bound for the $ERRA$ was set to 0. A further investigation that aims to produce a reliable and efficient method to find a tighter initial lower bound could be worth the effort. This paper also introduces the variable $\alpha-$neighbour $p-$centre problem by using a simple scheme to set coverage requirements for individual demands. From a decision maker's view point, this approach can be applied in practice by setting coverage requirements according to data distribution or the importance ranking of customers that may be available for consideration. Finally, $ERRA$ can also be adapted to other related

location problems, such as the constrained continuous $p-$centre problem with the presence of forbidden regions.

Acknowledgments

The first author would like to thank EPSRC for her PhD studentship.

References

Al-Khedhairi, A. & Salhi, S. (2005). Enhancements to Two Exact Algorithms for Solving the Vertex $P-$Centre Problem. Journal of Mathematical Modelling and Algorithms, 4, 129-147.

Barber, C, Dobkin, D. & Huhdanpaa, H., (1996). A Quickhull Algorithm for Convex Hulls. ACM Transactions on Mathematical Software, 22, 469-483.

Callaghan, B. (2016). An Investigation into Exact Methods for the Continuous $p-$Centre Problem and its Related Problems. Ph.D. Thesis, University of Kent at Canterbury, UK.

Callaghan, B., Salhi, S., & Nagy, G. (2017). Speeding up the Optimal Method of Drezner's for the $p-$Centre Problem in the Plane. European Journal of Operational Research, 257, 722-734.

Chen, D., & Chen, R. (2009). New Relaxation-based Algorithms for the Optimal Solution of the Continuous and Discrete Problems. Computers and Operations Research, 36, 1646-1655.

Chen, D., & Chen, R. (2010). A relaxation based algorithm for solving the conditional $p-$center problem. Operations Research Letters, 38, 215-217.

Chen, D., and Chen, R. (2013). Optimal Algorithms for the $\alpha-$Neighbor $P-$Center Problem. European Journal of Operational Research, 225, 36-43.

Chen, R., & Handler, G.Y. (1987). Relaxation Method for the Solution of the Minimax Location-Allocation Problem in Euclidean Space. Naval Research Logistics, 34, 775-788.

Chen, R., & Handler, G. Y. (1993). The Conditional $P-$Center Problem in the Plane. Naval Research Logistics, 40, 117-127.

Cooper, L. (1961). Location-Allocation Problems. Operations Research, 3, 331-343.

Cooper, L. (1964). Heuristic Methods for Location-Allocation Problems. SIAM Review, 6, 37-53.

Daskin, M. S, (1995), *Network and discrete location: Models, algorithms and applications,* John Wiley & Sons.

Drezner, Z. (1984). The $p-$centre problem Heuristic and Optimal Algorithms. Journal of the Operational Research Society, 35, 741-748.

Drezner, T., Drezner, Z., & Salhi, S., (2006). A multi-objective heuristic for the casualty collection points location problem. Journal of the Operational Research Society, 57, 727-734.

Drezner, Z. (1989). Conditional $p-$center problems. Transportation Science, 23, 51-53.

Dongarra, J. J. (1992). Performance of various computers using standard linear equation software. ¡http://www.netlib.org/benchmark/performance.pdf¿.

Elshaikh, A., Salhi, S. & Nagy, G. (2015). The continuous $p-$centre problem: An investigation into variable neighbourhood search with memory. European Journal of Operational Research, 241, 606-621

Elshaikh, A., Salhi, S., Brimberg, J., Mladenović, N., Callaghan, B., & Nagy, G., (2016). Adaptive Perturbation-Based Heuristics: An Application for the Continuous $p-$Centre Problem. Computers & Operations Research, 75, 1-11.

Elzinga, J., and Hearn, D. (1972). Geometric Solutions for some Minimax Location Problems. Transportation Science, 6, 379-394.

Handler, G.Y. & Mirchandani, P.B. (1979). Locations on Networks: Theory and Algorithms. Cambridge, MA: MIT Press.

Irawan, C., Salhi, S. & Drezner, Z. (2016). Hybrid meta-heuristics with VNS and exact methods: application to large unconditional and conditional vertex $p-$centre problems. Journal of Heuristics, 22, 507-537.

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, 867-877.

Khuller, S., Pless, R., & Sussmann, Y. J. (2000). Fault tolerant $K-$center problems. Theoretical Computer Science, 242, 237-245.

Krumke, S.O. (1995). On a generalization of the $p-$centre problem. Information Processing Letters, 56, 67-71.

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

Minieka, E. (1970). The $m-$centre problem. SIAM Review, 12, 138-139.

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, 3075-3091.

Toregas, C. (1971). Location Under Maximal Travel Time Constraints. Ph.D. Dissertation, Cornell University.

Wei, H, Murray., A. T., & Xiao, N. (2006). Solving the continuous space $p-$center problem: planning application issues. IMA Journal of Management Mathematics, 17, 413-425.

Sylvester, J.J. (1860). On Poncelet's approximate valuation of surd forms. Philos Mag 20, 203-222.

Appendix

		CPU Time (secs)		
p	Z^*	Without Enhancements$^{\perp}$, $k = 3$	With AddE2$^{\perp}$, $k = 2$	With SubE1 & AddE2$^{\perp}$, $k = 3$
10	1716.510	4.18	2.08	1.81
20	1029.715	3.22	4.63	2.32
30	739.193	3.70	10.51	5.94
40	580.005	38.69	24.87	49.97
50	468.542	63.62	83.76	53.08
60	400.195	24.49	20.76	12.10
70	357.946	57.12	51.32	47.07
80	312.500	41.62	29.32	47.43
90	280.903	63.24	38.33	48.20
100	256.680	17.32	17.81	13.36
Average	**614.22**	**31.73**	**28.34**	**28.13**

$^{\perp}$ Best result for $k = 1, \ldots, 10$.

Table A.1: Results for the Reverse Relaxation Algorithm without enhancements, with AddE2 and with SubE1 & AddE2 where $n = 439$

	$\alpha = 3$			
p	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	3989.302	0.49	29	34
20	2347.505	6.27	76	114
30	1716.510	2.01	61	93
40	1407.624	3.21	98	91
50	1226.020	3.23	93	81
60	1019.986	5.76	110	150
70	946.457	17.48	141	215
80	853.028	54.00	172	275
90	739.193	21.18	165	211
100	657.885	77.691	186	384
Average	**1490.35**	**19.05**	**113**	**164**

Table A.2: Results for the 3−neighbour p−centre problem for $pr439$

	$\alpha = 2$				$\alpha = 3$			
p	Z^*	CPU (secs)	Sub_{Max}	# Iterations	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	114.064	13.29	71	55	137.023	1.77	41	37
20	67.926	110.95	118	111	89.885	67.00	116	80
30	$54.073^{\perp}$	86400.00	235	287	67.926	87.72	121	158
40	$44.782^{\perp}$	86400.00	258	295	58.592	15674.13	238	468
50	$39.061^{\perp}$	86400.00	269	267	$49.941^{\perp}$	86400.00	245	483
60	$34.599^{\perp}$	86400.00	274	235	$44.406^{\perp}$	86400.00	228	363
70	$31.099^{\perp}$	86400.00	281	171	$40.736^{\perp}$	86400.00	261	508
80	$28.513^{\perp}$	86400.00	298	183	$37.207^{\perp}$	86400.00	276	527
90	$26.727^{\perp}$	86400.00	335	221	$34.558^{\perp}$	86400.00	295	518
100	$25.008^{\perp}$	86400.00	346	198	$32.070^{\perp}$	86400.00	284	426
Average	$- - -$	**69132.42**	**249**	**202**	$- - -$	**53423.06**	**210**	**356**

$^{\perp}$ Best lower bound found in 86400.00 secs (lower bound)

Table A.3: Results for the $\alpha-$neighbour $p-$centre problem for $rat575$ and $\alpha = 2$ & 3

	$\alpha = 2$				$\alpha = 3$			
p	Z^*	CPU (secs)	Sub_{Max}	# Iterations	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	131.846	20.84	79	73	160.721	0.87	33	32
20	79.312	87.30	121	118	106.304	171.29	126	118
30	$63.343^{\perp}$	86400.00	247	373	79.313	248.15	139	202
40	$52.131^{\perp}$	86400.00	268	314	$69.065^{\perp}$	86400.00	236	796
50	$45.868^{\perp}$	86400.00	273	238	$58.831^{\perp}$	86400.00	257	898
60	$40.336^{\perp}$	86400.00	302	264	$51.886^{\perp}$	86400.00	246	1073
70	$36.433^{\perp}$	86400.00	283	173	$47.383^{\perp}$	86400.00	249	860
80	$33.634^{\perp}$	86400.00	318	198	$43.276^{\perp}$	86400.00	254	747
90	$31.117^{\perp}$	86400.00	321	154	$40.083^{\perp}$	86400.00	295	773
100	$29.300^{\perp}$	86400.00	352	165	$37.314^{\perp}$	86400.00	270	632
Average	$- - -$	**69130.81**	**256**	**207**	$- - -$	**60522.03**	**210**	**613**

$^{\perp}$ Best lower bound found in 86400.00 secs (lower bound)

Table A.4: Results for the $\alpha-$neighbour $p-$centre problem for $rat783$ and $\alpha = 2$ & 3

	$\alpha = 2$				$\alpha = 3$			
p	Z^*	CPU (secs)	Sub_{Max}	# Iterations	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	3641.566	1.76	51	34	5268.064	2.96	45	39
20	2389.356	193.89	128	127	3107.089	21.83	93	75
30	1916.540	72557.25	262	396	2389.356	123.40	131	125
40	$1568.679^{\perp}$	86400.00	268	243	2038.961	45414.38	260	385
50	1347.869	3525.44	293	217	$1760.26^{\perp}$	86400.00	251	328
60	$1221.289^{\perp}$	86400.00	332	249	$1563.75^{\perp}$	86400.00	292	301
70	$1110.214^{\perp}$	86400.00	351	247	$1430.211^{\perp}$	86400.00	307	311
80	$1011.461^{\perp}$	86400.00	420	267	$1290.213^{\perp}$	86400.00	302	233
90	$940.597^{\perp}$	86400.00	383	232	$1221.17^{\perp}$	86400.00	357	276
100	$866.083^{\perp}$	86400.00	360	176	$1135.70^{\perp}$	86400.00	336	240
Average	$- - -$	**59467.83**	**285**	**218**	$- - -$	**56396.26**	**237**	**231**

$^{\perp}$ Best lower bound found in 86400.00 secs (lower bound)

Table A.5: Results for the $\alpha-$neighbour $p-$centre problem for $pr1002$ and $\alpha = 2$ & 3

	$\alpha = 2$				$\alpha = 3$			
p	Z^*	CPU (secs)	Sub_{Max}	# Iterations	Z^*	CPU (secs)	Sub_{Max}	# Iterations
10	4441.013	8.56	67	52	6200.039	2.85	46	39
20	$2882.010^{\perp}$	86400	237	433	3716.203	40.39	144	110
30	$2222.236^{\perp}$	86400	259	452	$2880.940^{\perp}$	86400.00	252	430
40	$1827.67^{\perp}$	86400	279	471	$2372.675^{\perp}$	86400.00	264	662
50	$1587.654^{\perp}$	86400	299	335	$2090.673^{\perp}$	86400.00	287	783
60	$1428.493^{\perp}$	86400	324	321	$1809.207^{\perp}$	86400.00	264	675
70	$1275.304^{\perp}$	86400	335	213	$1648.718^{\perp}$	86400.00	293	935
80	$1173.849^{\perp}$	86400	378	252	$1503.291^{\perp}$	86400.00	296	637
90	$1087.53^{\perp}$	86400	365	216	$1417.600^{\perp}$	86400.00	293	693
100	$1009.66^{\perp}$	86400	394	205	$1326.82^{\perp}$	86400.00	334	807
Average	$- - -$	**77760.86**	**294**	**297**	$- - -$	**69124.32**	**247**	**577**

$^{\perp}$ Best lower bound found in 86400.00 secs (lower bound)

Table A.6: Results for the $\alpha-$neighbour $p-$centre problem for $rl1323$ and $\alpha = 2$ & 3