Speeding up the Optimal Method of Drezner's for the $p-$Centre Problem in the Plane*

Becky Callaghan, Said Salhi, Gábor Nagy

Centre for Logistics & Heuristic Optimisation (CLHO), Kent Business School, The University of Kent, Canterbury, Kent, UK, CT2 7NZ

Abstract

This paper revisits an early but interesting optimal algorithm first proposed by Drezner to solve the continuous $p-$centre problem. The original algorithm is reexamined and efficient neighbourhood reductions which are mathematically supported are proposed to improve its overall computational performance. The revised algorithm yields a considerably high reduction in computational time reaching, in some cases, a decrease of 96%. This new algorithm is now able to find proven optimal solutions for large data sets with over 1300 demand points and various values of p for the first time.

Keywords: Location, $p-$centre problem, continuous space, $Z-$maximal circles, optimal algorithm.

1. Introduction

The $p-$centre problem seeks to minimise the maximum distance or travel time whilst ensuring all the n demand points are covered by at least one of the p chosen facilities. This problem can be categorised as either the vertex $p-$centre problem or the absolute $p-$centre problem. In the former, which is the discrete case, the optimal facilities are part of a set of the potential facility sites which can be either the demand points or other known sites. However, in the latter the facilities can be located anywhere along network edges (as introduced by not solved by Hakimi (1965)) or in the plane.

In this paper, we will explore the absolute $p-$centre problem in the plane, which is also

*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.

Email addresses: bc349@kent.ac.uk (Becky Callaghan), s.salhi@kent.ac.uk (Said Salhi), g.nagy@kent.ac.uk (Gábor Nagy)

known as the continuous or the planar p-centre problem. It is worth noting that the continuous $p-$centre problem, besides being used for interesting real life location applications that will be briefly mentioned next, could also provide a greenfield solution which can be used as a guide to identify potential sites for the discrete case as in some cases this data can be very expensive to gather. In addition, the $p-$centre problem can also be used as a basis for academic research in the general area of global optimisation including other continuous related location problems.

Here are some of the papers describing real-life problems tackled by $p-$centre models. One of the earliest applications is by Richard, Beguin & Peeter's (1990) who used the $p-$centre problem to locate fifteen fire stations in the Belgian province of Luxembourg. Pacheco & Casado (2004) located a number of health resources such as geriatric and diabetic health care clinics in the rural area of Burgos in Spain. Wei *et al.* (2006) adapted their Voronoi-based algorithm developed for the constrained continuous $p-$centre problem to locate twenty-five emergency warning sirens in Dublin, Ohio. Kaveh & Nasr (2011) modified a harmony search heuristic to locate bicycle stations in Isfahan, Iran by solving the conditional and unconditional discrete $p-$centre problem. Finally, Lu (2013) used the $p-$centre problem to locate a number of urgent relief distribution centres after the 7.3 Richter scale earthquake in Taiwan.

Most of the real-life applications for the $p-$centre problem have been solved successfully using powerful heuristics and metaheuristics. However, recent developments in exact methods, with the advances in computing power, memory management and powerful commercial optimisation software such as IBM ILOG CPLEX, mean that the proven optimal solution can now be worth exploring for larger problems. This study aims to respond to such scientific and technological change. In addition, if an optimal solution can be found in a reasonable amount of time, this will provide flexibility in performing scenario analysis for strategic planning purposes which is of extreme importance in practice due to the massive investment usually required.

A Brief Literature Review

The single facility minimax location problem ($1-$centre) in the continuous space has a long history and was posed originally in 1857 by the English mathematician James Joseph Sylvester (1814-1897). A few years later, in 1860, he proposed an algorithm to solve it. Elzinga & Hearn (1972) developed an efficient and widely used geometrical-based algorithm

to solve the problem optimally. Their algorithm was adapted and enhanced by many authors including Elshaikh *et al.*(2015). For more information on continuous centre problems and references therein, see Drezner (2011). The idea was extended to find solutions to multi-facility location problems including the $p-$centre problem. Hakimi (1965) was the first to formulate the continuous $1-$centre problem in a network, and Minieka (1970) studied the case where $p > 1$. The first paper discussing the $p-$centre problem in the plane was by Chen (1983). The problem has been shown to be NP-hard when p is variable, see Megiddo & Supowit (1984). For a fixed value of p the problem can be solved in polynomial time, $O(n^{2p+4})$, though requiring an excessive amount of computational effort especially for larger values of p, see Drezner (1984).

There exist a few variations of the continuous $p-$centre problem. For example, Chen & Handler (1993) proposed an efficient algorithm to solve the conditional $p-$centre problem. Here, the aim is to locate p facilities given that q facilities already exist. Wei *et al.*(2006) suggested a Voronoi-based algorithm to solve the constrained continuous $p-$centre problem where the facilities cannot be located within some forbidden regions such as rivers, lakes, military areas etc. Chen & Chen (2013) used Minieka's algorithm and the relaxation method to solve the $\alpha-$neighbour $p-$centre problem. In this variation, each demand point is covered by at least α facilities which can be important in the case of facility disruption.

Among the most recent theoretical work is the use of the relaxation concept, where a large problem is broken down into relatively much smaller and more manageable sub$-$problems that are easier to solve. For more details on this particular topic, see Chen & Handler (1987), Chen & Chen (2009) and Chen & Chen (2010). For the discrete case, though not directly related to our research, the following studies by Elloumi *et al.*(2004), Brandenberg & Roth (2009) and Caruso *et al.*(2003) can be found to be interesting and also informative. In both the discrete and the continuous problems, Cooper's (1964) Multi-Start method, which is based on the locate-allocate principle, is often used to produce an upper bound for optimal methods or initial solutions for metaheuristics.

This paper will be analysing the original continuous $p-$centre problem by revisiting an interesting, though originally very slow, optimal algorithm proposed thirty years ago by Drezner (1984). This older method used a subset of facility locations based on specific circles rather than demand points. As this algorithm is the basis of our research, it is detailed in the next section.

The contributions of this study include:

i) revisiting an early but slow optimal algorithm for the continuous $p-$centre problem;

ii) introducing neighbourhood reduction schemes supported mathematically to improve drastically the computational performance of this exact method;

iii) embedding an adaptive CPLEX policy to further enhance its efficiency;

iv) solving optimally for the first time relatively much larger problems with up to 1300 demand points and up to 100 facilities.

The paper is organised as follows: the investigated exact method is introduced and described in Section 2, alongside initial results based on the original algorithm. Section 3 proposes the suggested enhancements to the algorithm which are supported by new lemmas and proofs. The computational results are given in Section 4 followed by an adaptive CPLEX policy in Section 5 making this revised optimal algorithm even more efficient. The overall computational results are given in Section 6. Our conclusions and suggestions are summarised in the final section.

2. Drezner's Optimal Algorithm

2.1. Introduction

Drezner's algorithm is based on the idea of $Z-$*maximal circles*. A circle is defined as maximal based on a given upper bound, Z. The set of maximal circles based on Z is then identified and their respective centres are then used as a subset for the potential facility locations.

Let us define the following notations.

I: set of demand points indexed by $i = 1 \dots n$;

J: set of all possible circles indexed by $j = 1 \dots m$;

C_j: circle j defined by its centre (x_j^c, y_j^c) and radius r_j, $j \in J$;

K: subset of I;

$R(K)$: the radius of the smallest circle encompassing all points in K;

$d_{i,j}$: 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;

$d'_{i,l}$: Euclidean distance from demand point i to demand point l;

Z: the upper bound at a given iteration;

J_Z: set of $Z-$maximal circles ($J_Z \subset J$).

Definition 2.1. *The* **closure of circle** C_j *is the set of demand points encompassed by circle* C_j *which is defined as*

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

Definition 2.2. *The* **minimum covering circle** (MCC) *of the set* K *is the smallest circle encompassing all points in* K *with radius* $R(K)$.

We can now define a $Z-$maximal circle in the following way, as given by Drezner (1984).

Definition 2.3. *A circle* C_j *with radius* r_j *is said to be* $Z-$**maximal** *(often simply called maximal) if:*

1. $r_j < Z$;
2. *For every demand point* $i \notin Cl_j$, $R(Cl_j \cup \{i\}) \geq Z$.

Drezner proposed two ways to solve the $p-$centre problem using $Z-$maximal circles. The first, which will be referred to as $CP_0^{(a)}$, uses the set covering problem to find the minimum number of $Z-$maximal circles needed. First, let the input $A_{i,j}$ be defined as

$$A_{i,j} \begin{cases} 1 \text{ if } i \in Cl_j, \\ 0 \text{ else.} \end{cases}$$

$$(CP_0^{(a)}) \text{ Minimise } \sum_{j \in J_Z} x_j \tag{1}$$

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

$$x_j \in \{0,1\} \quad \forall j \in J_Z, \tag{3}$$

$$\text{where } x_j = \begin{cases} 1 \text{ if } Z-\text{maximal circle } C_j \text{ is selected,} \\ 0 \text{ else.} \end{cases}$$

The objective function (1) refers to minimising the number of $Z-$maximal circles. Constraint (2) guarantees that every demand point is encompassed, or covered, by at least one $Z-$maximal circle.

In the second method, referred to as $CP_0^{(b)}$, a new constraint (4) is added to $CP_0^{(a)}$ to impose that the number of covering circles has to be equal to p, while the objective function (1) is omitted turning the problem into a feasibility problem.

$$(CP_0^{(b)}): \text{ Find } x_j \in \{0,1\} \ \forall j \in J_Z$$

$$\text{subject to } (2)-(3),$$

$$\sum_{j \in J_Z} x_j = p. \tag{4}$$

If the minimum number of covering circles found in (1) is $\leq p$ or if $CP_0^{(b)}$ is feasible, then the upper bound is decreased by setting Z to the radius of the largest $Z-$maximal circle from the obtained solution, and the process of identifying the $Z-$maximal circles is then repeated. Otherwise (i.e. the minimum number is $> p$ or $CP_0^{(b)}$ is infeasible), the current upper bound Z is taken as the optimal solution and the algorithm terminates.

Before we use Drezner's optimal algorithm, as described in Figure 1, we shall first define the following additional notations.

C_J^1: the set of null circles created from one critical point only (i.e., note: $r_j = 0 \ \forall \ C_j \in C_J^1$);
C_J^2: the set of circles created from two critical points defining its diameter;
C_J^3: the set of circles made up from three critical points forming an acute triangle.

It is important to note that an appropriate heuristic could be used to find an initial up-

per bound in Step 2. For instance, a simple multi start heuristic can be used. In this study we opted for the H_2 heuristic proposed by Drezner (1984) for consistency reasons.

Step 1. Find all three sets of circles C_J^1, C_J^2 and C_J^3.
Step 2. Find an initial solution and set the solution value as the initial upper bound, Z.
Step 3. Eliminate all circles whose radii are $\geq Z$ from C_J^2 and C_J^3.
Step 4. Find all $Z-$maximal circles using Procedure $FMC(C_J^1, C_J^2, C_J^3, Z, J_Z)$ (see Figure 2).
Step 5. Solve $CP_0^{(a)}$ or $CP_0^{(b)}$ using the $Z-$maximal circles J_Z.
If the optimal value of $CP_0^{(a)}$ is $\leq p$ or $CP_0^{(b)}$ is feasible, set the new upper bound Z as the radius of the largest $Z-$maximal circle found in the solution and go to Step 3.
Else take the upper bound Z as the optimal solution value of the planar $p-$centre problem and stop.

Figure 1: Drezner's Original Algorithm (Drezner (1984))

Procedure $FMC(C_J^1, C_J^2, C_J^3, Z, J_Z)$
Step 1. Set $J_Z = \emptyset$.
Step 2. (Find all $Z-$maximal circles in the set C_J^1.)
For each $C_j = \{i\} \in C_J^1$ do: If $\frac{d'_{j,l}}{2} > Z \ \forall \ l \neq i \in I$, add C_J^1 to J_Z. Else the (single point) circle C_j is not $Z-$maximal.
Step 3. (Find all $Z-$maximal circles in the set $C_J^2 \cup C_J^3$.)
While $\exists \ C_j \in \{C_J^2 \cup C_J^3\}$ do:
If $R(Cl_j \cup \{i\}) > Z \ \forall \ i \notin Cl_j$ (i.e. the circle C_j is $Z-$maximal by definition) set $J_Z = J_Z \cup \{C_j\}$.
Else, the circle C_j is not $Z-$maximal.
Step 4. Return J_Z.

Figure 2: The FMC Algorithm given a threshold Z

2.2. Initial results & the need for an improved implementation

Our initial results were found for two TSP-Library (2015) data sets, namely $pr439$ and $rat575$ which represent a 439-city problem and a 575-rattled grid problem respectively. Note that the basic tricks of using squared distances were also adopted here when required to improve code efficiency (e.g. when distances are compared, or for non-acute triangle detection).

Both $CP_0^{(a)}$ and $CP_0^{(b)}$ were initially used to solve the $p-$centre problem, and both were

found to take a considerable amount of computational time as a large number of iterations was required. As an illustrative example, we show the result found for the TSP-Library data sets $pr439$ and $rat575$ where $p = 90$. For the data set $pr439$, the 90−centre problem was optimally solved using $CP_0^{(a)}$ requiring more than 38 hours (i.e. 137692.6 seconds) and 4580 iterations. When using $CP_0^{(b)}$, the time was reduced to just below 3 hours (10654.30 seconds) while using 393 iterations only. For $rat575$, an optimal result was obtained using $CP_0^{(b)}$, however it required nearly 30 hours (107916.0 seconds) and 2729 iterations. Furthermore, when using $CP_0^{(a)}$, the program was stopped after the time limit of 2 days with only one feasible solution found with a value of 21.471 (a percentage difference of 18% from the optimal solution). It will be shown later that the optimal solution can be found in less than half an hour (996.43 seconds) with our improved method. This example highlights the importance of developing ways to enhance the efficiency of Drezner's algorithm optimal algorithm.

2.3. Modification of the Covering Problem (Enhancement Zero)

Traditionally, the continuous $p-$centre problem is formulated as the Euclidean unweighted $p-$centre problem. This multiple facility location problem has been examined by a small number of authors, see Plastria (2002) and the references therein. It can also be formulated as a non-linear mathematical programming formulation. However, the formulation that we will use in this paper is similar to Drezner's $CP_0^{(b)}$ formulation with two commonly used additions consisting of a) an objective function that aims to minimise the largest radius (5) and b) an extra constraint to deal with the characteristics of the $p-$centre (8). This formulation, referred to as CP_1, will be used throughout this work.

$$(CP_1) \text{ Minimise } D \tag{5}$$

$$\text{subject to } \sum_{j \in J_Z} A_{i,j} x_j \geq 1 \quad \forall i \in I, \tag{6}$$

$$\sum_{j \in J_Z} x_j = p, \tag{7}$$

$$x_j r_j \leq D \quad \forall j \in J_Z, \tag{8}$$

$$x_j \in \{0, 1\} \ \forall j \in J_Z. \tag{9}$$

where

D: the maximum distance between a facility and a demand point.

The use of CP_1 is first tested on the previous two data sets. It was observed that for $p = 90$ and 100, the computational times were 1258 and 462 seconds respectively, approximately 9 (resp. 7) times faster than using $CP_0^{(a)}$ (resp. $CP_0^{(b)}$). It is also worth noting that CP_1 has an advantage over Drezner's original suggestions as the optimal solution value D is much tighter leading to requiring a relatively smaller number of iterations. Although it may be harder to solve CP_1 than $CP_0^{(a)}$ or $CP_0^{(b)}$, the last two require a large number of iterations, each including a lengthy $Z-$maximal circles identification step.

2.4. Observations

Optimal solutions for $p = 10, 20 \ldots, 100$ were found using CP_1 instead of $CP_0^{(a)}$ or $CP_0^{(b)}$ for the TSP-Library data sets $pr439$ and $rat575$. The results are given in the Appendix under Tables A.2 & A.3 and Figure A.1. Based on these results, it can be observed that there are two areas where enhancements could be introduced in an attempt to shorten the overall computational time. These include:

a) the way the $Z-$maximal circles are identified from one iteration to the next;
b) a choice of a compromise between the quality of a feasible solution and its corresponding computational time when solving CP_1 (i.e. finding an optimal solution or just a good feasible solution).

This paper will now investigate several ways in which the original algorithm using CP_1 can be efficiently implemented. Sections 3−5 will cover (a) and Section 6 will deal with (b).

Note that the introduction of CP_1, instead of using $CP_0^{(a)}$ or $CP_0^{(b)}$, could be considered as our first enhancement due to generating tighter bounds. However, for simplicity and conciseness, the results of CP_1 will be used as our starting point from which we will base our improvements.

3. The $Z-$maximal circles-Based Enhancements

3.1. Enhancement One: EHA-Based Implementation

The Elzinga-Hearn algorithm (EHA) is used to find the MCC of a set of demand points. As this is repeatedly needed in Step 3 of the FMC algorithm, in order to calculate $R(Cl_j \cup \{i\})$, two ways in which the overall time performance can be enhanced are highlighted.

Early Termination

The EHA starts with a circle made from any two selected points and continues to find a covering circle of increasing size until all points are covered. It is important to realise that in the FMC algorithm, the exact centre point and the radius of the MCC are not needed: we simply aim to establish whether or not the radius of the MCC will be larger or smaller than the upper bound Z.

If the MCC is smaller than the upper bound, then the EHA will continue until the end as normal. However, it can be terminated early if the circle's radius exceeds Z during the algorithm. This is because at each iteration in the EHA, the new circle's radius is either the same or larger. Therefore, if a circle has a radius $\geq Z$ at any point in the algorithm there is no need to continue as the final circle (the MCC) will be even larger.

More Informative Initial Points

Instead of starting the EHA from random points or selecting points using selection rules, such as the ones adapted by other authors including Welzl (1991) and Elshaikh *et al.*(2015), we take into account the information we have already found. In other words, the two or the three critical points that defined the circle found at a current iteration are the points that we choose as our initial points for the EHA. This makes the selection deterministic and yields faster results.

This double enhancement, referred to as Enh1, is incorporated only into Step 3 of the FMC algorithm and does not affect the total number of iterations of the algorithm.

3.2. Enhancement Two: Efficient Recording of the $Z-$maximal circles

At each new iteration in Drezner's algorithm, the process of finding the $Z-$maximal circles begins again from the start irrespective of earlier iterations. However, when examining the first set of results it was observed that many of the same circles were being classified as

$Z-$maximal during successive iterations.

As an example, Table 1 shows the number of $Z-$maximal circles found at each of the first 10 iterations of the original algorithm for the data set $pr439$ with $p = 100$. In this

Iteration #	# Original Circles	# $Z-$maximal circles	# Circles Previously Identified	Extra % Required
1	9281	860	-	-
2	9189	855	780	8.77
3	8835	797	597	25.09
4	8796	805	758	5.84
5	8652	809	684	15.45
6	8449	798	640	19.80
7	8384	804	735	8.58
8	7922	756	478	36.77
9	7855	767	693	9.64
10	7637	770	601	21.95
Average	8500.00	802.10	662.88	16.88

Table 1: Number of $Z-$maximal circles required & previously identified for the first 10 iterations $(n = 439, p = 100)$

example, approximately 17% of the new $Z-$maximal circles need to be identified at each iteration only, as the other ones have already been found in previous iterations. Therefore, a technique to identify whether a circle is $Z-$maximal or not in subsequent iterations is worthwhile constructing.

Lemma 1. *If circle C_j is Z_t-maximal at iteration t, then it is also $Z_{t+1}-$maximal for iteration $t+1$ if and only if its radius $r_j < Z_{t+1}$.*

Proof. We know at each iteration t, the upper bound Z strictly decreases. Therefore, we can say $Z_t > Z_{t+1}$. For circle C_j to be a $Z-$maximal circle at iteration t, the following two conditions need to be satisfied:

1. $r_j < Z_t$;
2. for every demand point $i \in I$ such that $i \notin Cl_j,\ R(Cl_j \cup \{i\}) \geq Z_t$.

As $Z_{t+1} < Z_t$, we can deduce that $R(Cl_j \cup \{i\}) > Z_{t+1}$. Thus if $r_j < Z_{t+1}$, circle C_j will still be a $Z-$maximal circle by definition at iteration $t+1$. □

The information denoting whether or not circle C_j has been found to be $Z-$maximal or not

can be stored in a binary or logical vector $CircMax$ where

$$CircMax_j = \begin{cases} 1 \text{ if } C_j \in J_{Z_t}, \\ 0 \text{ else.} \end{cases}$$

This result is incorporated into Steps 2 and 3 of the FMC algorithm to avoid performing redundant calculations. We will refer to this enhancement as Enh2.

3.3. Enhancement Three: Fast Identification of some Non-$Z-$maximal circles

This enhancement, which we will refer to as Enh3, aims to quickly identify some non-$Z-$maximal circles without performing unnecessary calculations. As an example, take circle C_j with a centre point (x_j^c, y_j^c) and radius $r_j < Z$. We can now create a new circle C_j^+ centered at (x_j^c, y_j^c) and with radius Z. Therefore, it is clear that $C_j \subset C_j^+$.

Lemma 2. *If $s \in I$ is not covered by C_j (i.e. $s \notin Cl_j$) but is strictly covered by C_j^+, then circle C_j is not $Z-$maximal.*

Proof. Let $s \in I$ with $s \notin Cl_j$ but strictly covered by Cl_j^+. Then the smallest circle, C, containing s and the whole circle C_j, contains all the points in Cl_j and is strictly contained in Cl_j^+. Hence, C's radius is at least $R(Cl_j \cup \{s\})$ and is strictly less than Z. It follows that $R(Cl_j \cup \{s\}) < Z$, and so C_j is not $Z-$maximal.

□

Thus a minimum distance, or threshold, of value Z is established. In other words, if there is at least one demand point not covered by circle C_j which lies within this distance, then the circle cannot be classified as $Z-$maximal.

In summary, if

$$\exists\ i \notin Cl_j \ \mid \ d_{i,j} < Z, \tag{10}$$

we can conclude that circle C_j is not $Z-$maximal.
Additionally, a maximum threshold of $2Z$ can also be added using Lemma 3.

Lemma 3. *Take any demand point $s \in I$ not covered by C_j. In case $d_{s,j} \geq 2Z$, then $R(Cl_j \cup \{s\}) > Z$.*

Proof. Take $s \in I$ with $d_{s,j} \geq 2Z$. Consider the circle C with centre s and radius $2Z$. As $r_j < Z$, the centre of C_j is not encompassed by C. Therefore, the circle arc of C_j lying within C is strictly less than half the circle.
But the critical points of C_j span at least half the circle, and so cannot all lie within C. Therefore, $\exists\ i \in Cl_j$ such that $d_{i,s} > 2Z$, which implies that $R(Cl_j \cup \{s\}) > Z$. □

Thus if a point that lies at a distance $\geq 2Z$ from (x_j^c, y_j^c) is added to the set of points encompassed by the circle C_j, the MCC that covers all these points would have a radius $\geq Z$. Thus, if this information is known, any point in this area does not need to be checked again and hence computational time can be saved without affecting the quality of the solution.

In summary, if

$$d_{i,j} \geq 2Z \ \ \forall\ i \notin Cl_j, \tag{11}$$

then we can conclude that circle C_j is $Z-$maximal.

These two observations lead to the construction of a *checking area* for circle C_j, say $Check_j$. This is represented by the shaded area in Figure 3, and is defined as follows:

$$Check_j = \{i \notin Cl_j \ \mid \ Z \leq d_{i,j} < 2Z\}. \tag{12}$$

We can therefore conclude that further calculations must be performed only if the two observations above are not true and $Check_j \neq \emptyset$.

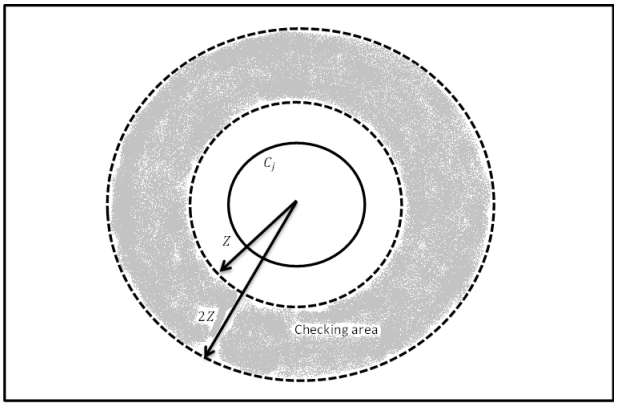


Figure 3: Checking Area for circle C_j

We incorporate Enh3 into Step 2 and Step 3 of the FMC algorithm.

3.4. Enhancement Four: Identifying Non-$Z-$maximal circles

If circle C_j is not maximal, then there must be a demand point $i \notin Cl_j$ such that $R(Cl_j \cup \{i\}) < Z$. If this point is recorded, in the next iteration this demand point can be the *first to be checked* and hence repeated computations can be discarded. If the MCC of the next iteration is still $< Z$, then we can deduce that this circle is still not $Z-$maximal thus saving computational time. If the MCC is $\geq Z$, we either continue with calculations and conclude it is now classified as $Z-$maximal, or we record the next demand point to cause C_j to be non-$Z-$maximal if it exists. In other words, either way will provide us with useful information that can be used in subsequent iterations.

As an example, say at iteration t it takes q_j points to find a demand point that determines circle C_j as not $Z-$maximal. This means the next iteration ought to start with the q_j^{th} point instead of starting from scratch at the beginning. This saves the computational time it takes to check the previous $(q_j - 1)$ points, say Sav_j^t. As this scheme is applied to C_j where $j = 1, \ldots, m'$, the saving at iteration t could be significant and of the order of $\sum_{j=1}^{m'} Sav_j^t$.

Let $Start$ be an integer vector of dimension m. The entry $Start_j$ denotes which demand point i should be checked first in the next iteration to see if circle C_j is $Z-$maximal or not.

This enhancement, referred to as Enh4, is incorporated into Step 2 and Step 3 of the FMC algorithm.

4. Analysing the $Z-$maximal circle-Based Enhancements

4.1. Individual Performances

The enhancements were first analysed separately so that each one's improvement in computational time could be assessed and its impact measured. For illustrative purposes, the computational times for the individual enhancements for the data set $pr439$ where $p = 70, 80, 90$ and 100 are first shown in Figure 4. This is then followed by combining all the refinements together using a certain order that will be based on the individual enhancement performances.

Figure 4 suggests that the best enhancement, giving an average decrease in computational time of 84.42%, is Enh3. By providing minimum and maximum thresholds by which the demand points are checked reduces many calculations as many points sit outside the checking area. Enh4 yields the second best result with an average decrease of 83.26% in

computational time. By starting at the last known non-$Z-$maximal circle all previous demand points can be disregarded, thus avoiding the unnecessary calculations that they incur. Enh1 is the third best at improving the overall computational time, with an average decrease of 50.65%. This enhancement reduces the number of calculations by terminating the EHA algorithm earlier whenever possible. Also, by choosing the current critical points as the initial points, the EHA will have less iterations to find the MCC. Finally, Enh2 improves the computational time the least. This is due to not dealing with the $Z-$maximal circle calculations directly; it simply minimises how many circles are needed for these calculations. The average improvement of computational time for Enh2 is 26.26%, which is still significant.

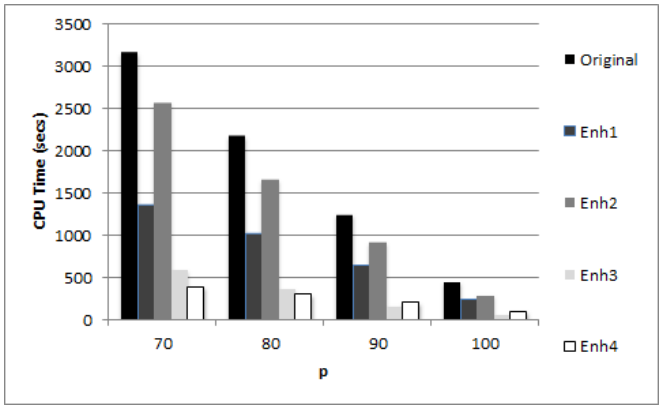


Figure 4: Individual Performances

4.2. Combined Performance

The four enhancements are embedded into Drezner's original algorithm that uses formulation CP_1. These are added in the order of individual performances observed earlier which is as follows: Enh3−Enh4−Enh1−Enh2. To assess the incremental gain of these enhancements we also conduct the following experiment: in the first run we use Enh3, in the second we use Enh3 and Enh4, and in the third Enh3, Enh4 and Enh1 are used. The fourth run consists of the overall algorithm with all the enhancements incorporated as noted earlier. The results are shown in Figure 5.

It is clear that the enhancements greatly improve the computational time. The first enhancement reduces the total computational time by an average of 84.49% as noted earlier, and by adding Enh4 this is decreased further to 90.26%. After the addition of Enh1, the average decrease becomes 96.46% and finally with all enhancements added this reaches a massive saving of 96.71%. In other words, just above 3% of computational time is really

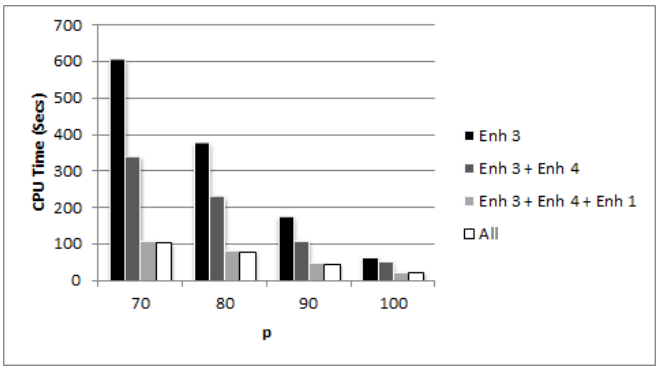


Figure 5: Comparison on CPU Time for the Enhancements

needed on average, leading to an exciting and strong result.

It is also worth noting that the incremental decrease in computational time is not directly additive as there is a high level of association between their individual contributions. For instance, after gaining 84% with Enh3, one might expect Enh4 to yield 83% of the remaining 16%. This would therefore give a new decrease of approximately 97%. However, it only decreases it to just over 90% (i.e., an extra 5.8% only).

4.3. The Complete Revised Optimal Algorithm

The revised FMC algorithm is given in Figure 6. It is similar to the original FMC algorithm except Step 2 and Step 3 in Figure 2 have been modified accordingly to accommodate the enhancements described in this study. The revised Drezner algorithm is similar to the Drezner's original algorithm stated previously in Figure 1, except that in Step 5 the formulation CP_1 is used instead of $CP_0^{(a)}$ or $CP_0^{(b)}$ and an extra step (Step 3 shown in Figure 7) has been added to accommodate the enhancements. For completeness, we reproduce the full revised optimal algorithm in Figure 7.

5. Computational Results

The proposed algorithm was coded in $C++$ on a HP Elitebook 8570w with 12GB of memory. The IBM ILOG CPLEX 12.6 console was incorporated into the program using default parameters.

Tables 2 and 3 show the results found for the data sets $pr439$ and $rat575$. The first column titled p shows the required number of facilities. The initial upper bound value, denoted by Z in column 2, was found from a 1000 iteration runs of the H_2 heuristic described in Drezner (1984). The next column, titled Z^*, shows the optimal solution value, followed by

Step 1: Input vectors $Start_j$ and $CircMax_j$. Set $J_Z = \emptyset$.

Step 2: For all $C_j \in J$, if $CircMax_j = 1$, add C_j to J_Z (i.e. set $J_Z = J_Z \cup C_j$) and remove C_j from the set $\{C_J^1 \cup C_J^2 \cup C_J^3\}$.

Step 3: (Find all $Z-$maximal circles in the set C_J^1.)
For all $C_j \in C_J^1$ where $CircMax_j = 0$ do:
Take the circle C_j^1. Starting from demand point $Start_j$, take $l \in I$ and compute $d'_{j,l}$. If $\exists\ l \in I$ such that $\frac{d'_{j,l}}{2} < Z$, the circle is not $Z-$maximal. Discard C_j from further investigation and set $Start_j = l$.
Else, the circle is $Z-$maximal by definition. Add C_j to J_Z (i.e. $J_Z = J_Z \cup C_j$) and set $CircMax_j = 1$.

Step 4: (Find all $Z-$maximal circles in the set $C_J^2 \cup C_J^3$.)
For all $C_j \in \{C_J^2 \cup C_J^3\}$ where $CircMax_j = 0$ do:
Step $4A$:

(i) Starting from demand point $Start_j$, if $\exists\ i \notin Cl_j$ where $d_{i,j} < Z$, go to Step $4B(ii)$, else go to $4A(ii)$.

(ii) Starting from demand point $Start_j$, if $\forall\ \ i \notin Cl_j,\ d_{i,j} \geq 2Z$, go to Step $4B(i)$, else go to $4A(iii)$.

(iii) While $\exists\ i \notin Cl_j$ with $Z \leq d_{i,j} < 2Z$, starting from demand point $Start_j$ do:
Use the EHA to find $R(Cl_j \cup \{i\})$.
If a circle with radius $\geq Z$ is found at any point during the EHA, go back to start of $4A(iii)$ starting from the next i value, else continue to find $R(Cl_j \cup \{i\})$. If $R(Cl_j \cup \{i\}) < Z$, then C_j is not $Z-$maximal: go to Step $4B(ii)$.

(iv) If this point is reached the circle is maximal: go to Step $4B(i)$.

Step $4B$:

(i) (Circle C_j is $Z-$maximal by definition.) Add C_j to J_Z (i.e. set $J_Z = J_Z \cup C_j$) and set $CircMax_j = 1$.

(ii) (Circle C_j is not $Z-$maximal by definition.) Set $Start_j = i$.

Figure 6: The FMC-Revised Algorithm

the computational time (in secs) required for the revised Drezner optimal algorithm to find Z^* in the Loop CPU Time column. Note that this result excludes the computational time consumed by the H_2 heuristic.

Other information, such as how many loops (iterations) are needed to get the optimal solution value, the total time spent on computing the $Z-$maximal circles and the total time spent on computing the result in CPLEX are reported alongside their corresponding percentages in the remaining columns. (Note that these two individual percentages when added are below 100% due to other calculations.)

For completeness, we also produced a summary result in Table 4 to show for both in-

Step 1. Find all circles made from one, two or three demand points. This creates three sets of circles : C_J^1, C_J^2 and C_J^3. Discard any circle in C_J^3 whose three points create an obtuse or right-angled triangle.

Step 2. Find an initial solution and set the solution value as the initial upper bound, Z.

Step 3. Set $Start_j = 1$ and $CircMax_j = 0$ for $j = 1, \ldots, m$.

Step 4. Eliminate all circles whose radii are $\geq Z$ from C_J^2 and C_J^3.

Step 5. Find all $Z-$maximal circles using the FMC-Revised algorithm with the threshold Z (Figure 6). Let J_Z be the set of $Z-$maximal circles.

Step 6. Solve CP_1 using the set of current $Z-$maximal circles in J_Z. If a solution is found, set Z to be the new upper bound, $J_Z = \emptyset$ and go back to Step 4.
Else, the upper bound Z is the optimal solution value of the planar $p-$centre problem and stop.

Figure 7: The Revised Drezner Optimal Algorithm

stances and for each value of p the new and the old duration including the percentage decrease. It is clear to see that the enhanced method has greatly reduced the computational time for both data sets. As an example, it took just over 4 hours average computational time for the data set $pr439$ previously, whereas now the average time is just over 12 minutes leading to a massive average reduction of 96%. Note that these computational times do not include the computational time for the H_2 heuristic.

For the data set $rat575$, the computational time has also been reduced. For the smaller values of p (10, 20 and 30), the majority of the time was taken computing the $Z-$maximal circles leading to a reduction of over 90%. However, for the other values of p the majority of the computational time is taken up solving the problem in CPLEX leading to an overall relatively small though still significant reduction of nearly 50%. This observation led us to face a challenge that will be explored in the next section.

Furthermore, our findings could be compared to the relaxation-based algorithms of Chen & Chen (2009) for the only reported results for the TSP-Library data set $pr439$. In this particular instance, our total computational time (inclusive of the computational time required for the H_2 heuristic) is found to be greater than theirs. However, it is also important to note that our optimal algorithm is deterministic and hence relatively more robust, as it is not sensitive to several factors including the initial subset of demand points or the number of demand points added to the subset at each iteration.

	H_2 Heuristic		Optimal Solution						
p	Z	CPU Time (secs)	Z^*	Loop CPU Time (secs)[a]	# Loops	Max Circles (secs)	CPLEX (secs)	Max Circles (%)	CPLEX (%)
10	1716.510	96.88	1716.510	342.78	2	278.96	34.52	81.38	10.07
20	1169.540	170.28	1029.715	2856.38	36	359.05	282.05	12.57	9.87
30	975.000	205.36	739.193	2146.67	49	229.60	207.87	10.70	9.68
40	874.271	218.9	580.005	1515.29	67	171.14	200.49	11.29	13.23
50	580.005	235.61	468.542	159.49	38	21.90	51.09	13.73	32.04
60	570.088	246.86	400.195	170.38	48	23.24	53.20	13.64	31.22
70	503.271	256.30	357.946	97.63	47	13.77	36.71	14.11	37.60
80	467.039	300.01	312.500	73.52	52	9.61	31.62	13.07	43.02
90	391.511	276.20	280.903	38.01	48	4.71	20.85	12.39	54.86
100	315.486	332.53	256.680	16.77	32	1.50	11.06	8.93	65.93
Average	756.272	233.90	614.218	741.69	42	111.35	92.95	19.18	30.75

Table 2: $n = 439$ TSP-Lib with Enhancements
[a] This excludes computational time for the H_2 heuristic.

	H_2 Heuristic		Optimal Solution						
p	Z	CPU Time (secs)	Z^*	Loop CPU Time (secs)[a]	# Loops	Max Circles (secs)	CPLEX (secs)	Max Circles (%)	CPLEX (%)
10	69.426	98.34	67.926	5572.02	10	693.86	336.28	12.45	6.04
20	48.107	175.62	45.475	1616.05	11	109.75	495.80	6.79	30.68
30	39.655	238.26	35.556	1023.14	14	46.20	544.21	4.51	53.19
40	33.365	296.90	30.063	37660.80	11	17.41	37514.80	0.05	99.61
50	30.336	403.76	25.826	6352.86	15	12.85	6247.59	0.20	98.34
60	27.951	422.18	23.163	26870.00	18	9.26	26800.50	0.03	99.74
70	25.578	558.85	20.858	26123.80	19	6.22	26082.30	0.02	99.84
80	24.135	535.90	19.026	32343.20	17	4.41	32343.20	0.01	99.91
90	21.932	743.20	17.460	2167.610	18	3.04	2149.99	0.14	99.19
100	20.402	795.13	16.420	25074.40	15	1.93	25074.40	0.01	99.95
Average	34.089	426.81	30.177	16480.39	15	90.49	15.758.90	2.42	78.65

Table 3: $n = 575$ TSP-Lib with Enhancements
[a] This excludes computational time for the H_2 heuristic.

	$pr439$			$rat575$		
p	Original CPU Time (secs)[a]	New CPU Time (secs)[a]	Percentage Decrease (%)	Original CPU Time (secs)[a]	New CPU Time (secs)[a]	Percentage Decrease (%)
10	6252.72	342.78	94.52	83898.60	5572.02	93.36
20	56753.00	2856.38	94.97	19087.6 0	1616.05	91.53
30	37017.10	2146.67	94.20	9743.91	1023.14	89.50
40	31355.00	1515.29	95.17	41733.00	37660.80	9.76
50	4939.25	159.49	96.77	9612.60	6352.86	33.91
60	4956.45	170.38	96.56	28344.00	26870.00	5.20
70	3170.89	97.63	96.92	40256.90	26123.80	35.11
80	2186.27	73.52	96.64	40181.70	32343.20	19.51
90	1258.22	38.01	96.98	4260.10	2167.61	49.12
100	462.30	16.77	96.37	33694.00	25074.40	25.58
Average	**14835.1197**	**741.6913**	**95.91**	**31081.242**	**16480.39**	**45.26**

Table 4: Original vs. Revised Drezner's algorithm for $n = 439$ TSP-Lib and $n = 575$ TSP-Lib
[a] This excludes computational time for the H_2 heuristic.

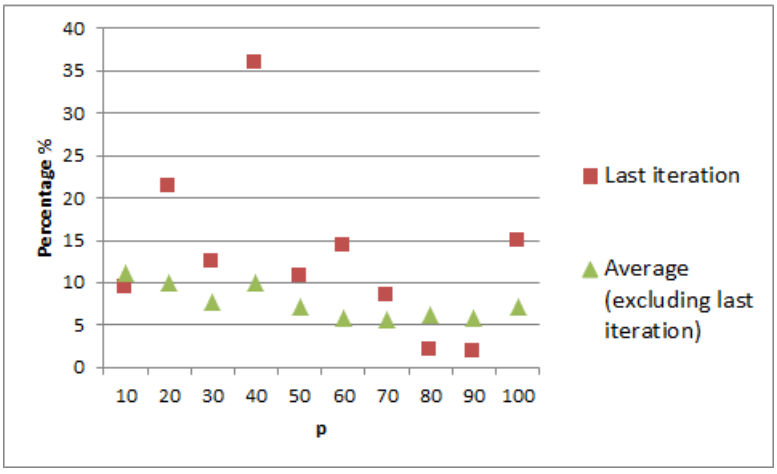


Figure 8: Average computational time % in CPLEX per iteration vs. last iteration for $rat575$

6. A Self-Adaptive CPLEX policy

In this section, we investigate how to balance the time spent between computing the $Z-$maximal circles and the level of the solution quality which we consider to be acceptable when solving CP_1. However, to guarantee optimality, we need to show at one stage that CP_1 has no feasible solution and hence the final iteration needs to run to the very end. In other words, it is not possible to reduce the computational time by terminating the search earlier in the last run.

Table 5 shows the total time taken in CPLEX compared to the time consumed in the last iteration in CPLEX. Though a relatively considerable amount of time is used in the last iteration accounting for approximately 10-20% of the total computational time, the computational time taken in the previous iterations is nonetheless worth exploring for possible improvement. A compromise feasible solution to save computational time in CPLEX while limiting the total number of iterations of the entire algorithm will be our focus in this section.

p	CPLEX Loop CPU Time (secs)	CPLEX Final Iteration CPU Time (secs)	Percentage Use (%)	# Loops	Average(%) per Loop excluding last iteration	Overall Average (%) per Loop
10	336.28	31.81	9.46	10	11.11	10.00
20	495.80	105.96	21.37	11	10.00	9.09
30	544.21	68.25	12.54	14	7.69	7.14
40	37514.80	12789.40	35.81	11	10.00	9.09
50	6247.59	673.42	10.78	15	7.15	6.67
60	26800.50	3821.60	14.26	18	5.88	5.56
70	26082.30	2231.55	8.56	19	5.56	5.26
80	32343.20	647.03	2.00	17	6.25	5.88
90	2149.99	41.48	1.93	18	5.88	5.56
100	25074.40	4577.57	18.26	15	7.14	6.67
Average	**15758.91**	**2498.81**	**15.86**	**15**	**7.14**	**6.67**

Table 5: CPLEX Durations (secs) for both the total and the last iteration in the case of $n = 575$ TSP-Lib

There are several ways in which the search can be terminated early in previous runs whilst producing a feasible solution for CP_1. An example would be to impose a time limit, however this does not always guarantee that a feasible solution will be found within that time and so other options are investigated.

Our study adopts a strategy by which we manipulate the duality gap so that CPLEX terminates earlier with a good feasible, but not necessarily optimal, solution whenever it manages to find at least one. However, the value of the duality gap can be both sensitive and critical which can make our algorithm less robust. The algorithm cannot terminate too

early as it could simply increase the number of iterations greatly, and therefore increase the time spent computing the $Z-$maximal circles. It is therefore important to find a reasonable compromise that we wish to devise. In this study, we propose the following self−learning CPLEX policy which takes into consideration information from previous iterations.

It is worth noting that the following duality gap policy is only implemented when CPLEX finds at least one feasible solution in any run of CPLEX. However, if no feasible solution has been identified in a given run, CPLEX continues until the maximum time limit is reached where the search terminates. Hence the obtained Z value of the previous run is used as the final solution, which obviously cannot be guaranteed to be optimal.

An Adaptive CPLEX Policy

At iteration t, the moving average for the computational time for calculating $Z-$maximal circles (T_{Max}) and solving the problem in CPLEX (T_{CPLEX}) based on the last α iterations is respectively defined as follows.

$$G_t^{\alpha}(A) = \frac{\sum_{t'=t-\alpha}^{t} A^{t'}}{\alpha} \tag{13}$$

where $A = \{T_{Max}, T_{CPLEX}\}$, and $A^{t'}$ is the corresponding time at iteration t'.

We define α as

$$\alpha = \begin{cases} \frac{t}{2} \text{ if } t \geq K, \\ t \text{ else.} \end{cases}$$

In other words, the classical average is used if $t < K$, otherwise the moving average over half of the past iterations is adopted. In this study, we used $K = 6$ based on preliminary results.

We use the following scheme based on the performance ratio $\xi = \frac{G_t^{\alpha}(T_{Max})}{G_t^{\alpha}(T_{CPLEX})}$;

a) If

$$\xi \geq 1 \tag{14}$$

then the time for computing the $Z-$maximal circles is much larger than the time spent solving the problem in CPLEX. Therefore, the number of iterations need to be reduced as

much as possible, and so we set the duality gap to 0%.

b) However, if

$$\xi \leq 0.4 \tag{15}$$

then the majority of the computational time is spent solving the problem in CPLEX, and therefore we wish to exit CPLEX sooner with a feasible solution rather than seeking an optimal one, hence we set the duality gap to be 1%.

c) If ξ has any other value, then the computational times are considered to be more or less similar. In this case, we wish to reach a balance between finding the near optimal solution and leaving CPLEX early, hence we set the duality gap to be 0.5%.

In summary, the following conditions related to the duality gap are given.

$$\text{Duality Gap} = \begin{cases} 0 & \text{if } \xi \geq 1, \\ 0.5\% & \text{if } 0.4 < \xi < 1, \\ 1\% & \text{if } \xi \leq 0.4. \end{cases} \tag{16}$$

This policy, which uses adaptive learning, is less sensitive to the effect of the data's distribution on the computational time and therefore it is very reliable.

The final results for $rat575$, that include the results where the CPLEX adaptive policy is incorporated, are found in Table 6, displayed alongside the total computational time required to optimally solve this data set using the enhanced algorithm without the duality gap policy. This table also shows that the average decrease in computational time is now 72.91% from the original CPU times, and it has decreased a further 50.05% from this new computational time when incorporating the duality gap with the enhancements. This is a promising result and demonstrates that the CPLEX adaptive policy has a large and positive effect on the overall efficiency of this enhanced algorithm.

It is important to recognise that for some values of p, such as $p = 10$, the total duration could be slightly increased as in this instance the majority of time is spent computing $Z-$maximal circles. This is because in the first iteration, we do not know whether the majority of time will be spent on computing the $Z-$maximal circles or solving the problem in

	Optimal Solution								
p	Loop CPU Time w/o Duality Gap (secs)[a]	Loop CPU Time (secs)[a]	Percentage Decrease (%)	# Loops	Max Circles (secs)	CPLEX (secs)		Max Circles (%)	CPLEX (%)
						Total	Last Loop		
10	5572.02	5732.12	-2.86	10	690.69	340.37	32.89	12.05	5.93
20	1616.05	1634.74	-1.15	11	112.83	471.74	108.68	6.90	28.86
30	1023.14	1254.57	-22.62	30	58.88	730.55	69.97	4.69	58.23
40	37660.80	25949.90	31.10	15	19.55	25793.20	12936.20	0.08	99.40
50	6352.86	3161.59	50.23	23	14.17	3052.89	675.08	0.45	96.56
60	26870.00	9134.14	66.01	29	10.49	9063.42	3733.39	0.11	99.26
70	26123.80	15961.50	38.91	24	6.53	15920.30	2219.57	0.04	99.74
80	32372.30	5656.99	82.53	74	8.80	5619.18	642.85	0.16	99.33
90	2167.61	996.43	54.03	34	3.98	976.77	41.86	0.40	98.03
100	25086.30	12862.90	48.73	23	2.29	12850.30	4614.62	0.02	99.90
Average	**16484.48**	**8234.49**	**50.05**	**27**	**92.82**	**7481.87**	**2507.51**	**2.49**	**78.52**

Table 6: $n = 575$ TSP-Lib with Enhancements and Duality Gap Policy
[a] This excludes computational time for the H_2 heuristic.

CPLEX as CPLEX has not run yet. To respond to this issue, we have therefore set a duality gap of 0.5% for the first iteration.

7. Overall Computational Results

Our algorithm was tested on the TSP-Lib data sets $rat575$, $rat783$, $pr1002$ and $rl1323$. For information, the data set $rat783$ represents a 783-rattled grid problem, and the data sets $pr1002$ and $rl1323$ refer to a 1002 and 1323-city problem respectively. As we aim to obtain optimal solutions, we used the best known heuristic results from Elshaikh *et al.*(2015) as our initial upper bound. This deviates from the method previously used, where the initial upper bound was found using the simple H_2 heuristic whose solutions may be relatively loose and hence may require an unnecessarily larger overall computational time. Note also that the computational times given here do not include this heuristic step, but these times are recorded in Elshaikh *et al.*(2015).

As these data sets are very large, a maximum time limit of 24 hours was set for each value of p. If the algorithm happens to take longer than the cutoff time, the program is terminated and the upper bound at that time is recorded as the best feasible solution.

Tables $7 - 10$ are arranged similarly to the tables in Section 5, with the newly found optimal solutions highlighted in bold. However, extra information for the computational

time spent in CPLEX is provided. In order to establish how much computational time cannot be improved on (the last iteration) the column representing the time spent in CPLEX is now divided into two, with one half showing the total time spent in CPLEX and the other half showing how long the last iteration took in CPLEX. Therefore, in the instance where the algorithm reaches the maximum time limit, the result in the second half of this column may not be showing the time spent to reach optimality. However, in each of these circumstances, no further feasible solution was found in the final iteration (except for the case where $n = 783, p = 40$). Thus, this indicates that the solution found in the previous iteration may be the optimal solution.

Furthermore, in the instance where $n = 783$ and $p = 40$, a feasible solution was found but the duality gap policy value had not been reached. The program was therefore allowed to run for a further hour (with the solution found at this iteration as its new upper bound) to see if this solution could be improved. Again, no further feasible solution was found which shows that the last feasible solution could be optimal. This last feasible solution found is the one given in Table 8.

	Best Heuristic	Optimal Solution							
p	Z	Z^*	Loop CPU Time (secs)[a]	# Loops	Max Circles (secs)	CPLEX (secs)		Max Circles (%)	CPLEX (%)
						Total	Last Loop		
10	67.926	**67.926**	489.53	1	413.20	32.37	32.37	84.41	6.612
20	45.6212	**45.475**	384.79	3	49.50	272.52	107.70	12.86	70.82
30	35.556	**35.556**	87.16	1	11.19	68.84	68.84	12.83	78.99
40	30.265	**30.063**	20898.30	5	6.57	20880.01	13085.80	0.03	99.91
50	26.173	**25.826**	2476.32	10	4.35	2462.60	670.71	0.18	99.45
60	23.622	**23.163**	8888.40	12	3.03	8878.01	3749.88	0.03	99.88
70	21.059	**20.858**	16283.70	9	1.64	16277.80	2238.12	0.01	99.9
80	19.510	**19.026**	3893.66	13	1.45	3887.75	646.53	0.04	99.85
90	17.923	**17.460**	868.39	18	1.22	863.18	41.75	0.14	99.40
100	16.551	**16.420**	13268.80	8	0.55	13265.40	4626.44	0.00	99.97
Average			6753.90	8	49.27	6688.86	2526.81	11.05	85.49

Table 7: Solutions for $n = 575$ TSP-Lib using the Revised Drezner's Algorithm starting from Best Heuristic Value

[a] This excludes computational time for the heuristic step.

	Best Heuristic	Optimal (or Best) Solution							
p	Z	Z^*	Loop CPU Time (secs) [a]	# Loops	Max Circles (secs)	CPLEX (secs)		Max Circles (%)	CPLEX (%)
						Total	Last Loop		
10	79.313	**79.313**	5696.39	2	2918.48	978.14	402.57	51.23	17.17
20	53.441	**53.332**	2884.05	8	224.16	2410.67	400.08	7.77	83.59
30	42.395	**42.307**	21833.60	4	55.52	21714.00	13229.40	0.25	99.45
40	35.962	35.861*	86400.00	1	19.30	86380.00	86370.00	0.02	99.98
50	31.184	31.041*	86400.00	10	14.81	86355.50	33887.70$^{\perp}$	0.01	99.95
60	28.053	27.880*	86400.00	14	10.95	86365.10	80032.39$^{\perp}$	0.01	99.96
70	25.446	25.239*	86400.00	3	4.21	86381.60	39254.10$^{\perp}$	0.004	99.98
80	23.560	23.192*	86400.00	9	5.43	86384.24	1530.90$^{\perp}$	0.006	99.98
90	21.710	21.319*	86400.00	12	5.01	86384.30	54352.70$^{\perp}$	0.005	99.98
100	20.334	19.999*	86400.00	7	2.03	86387.10	50190.10$^{\perp}$	0.002	99.99
Average				7	325.99	62974.05	35964.00	5.94	90.00

Table 8: Solutions for $n = 783$ TSP-Lib using the Revised Drezner's Algorithm starting from Best Heuristic Value

[a] This excludes computational time for the heuristic step.

* best feasible solution found within 86400 seconds.

$\perp$ no feasible solution found in the last iteration within the time limit allowed.

	Best Heuristic	Optimal Solution							
p	Z	Z^*	Loop CPU Time (secs) [a]	# Loops	Max Circles (secs)	CPLEX (secs)		Max Circles (%)	CPLEX (%)
						Total	Last Loop		
10^+	2389.360	–	–	–	–	–	–	–	–
20	1609.540	**1607.530**	4904.66	10	825.07	2786.07	340.83	16.82	56.80
30	1231.360	**1231.360**	881.26	1	86.42	739.83	739.83	9.81	83.95
40	1030.400	**1021.410**	1778.08	29	121.62	1404.82	190.49	6.84	79.01
50	901.455	**895.342**	13011.90	12	42.29	12867.60	353.84	0.33	98.89
60	801.474	**795.709**	8961.03	22	40.29	8843.69	785.27	0.45	98.69
70	727.154	**725.431**	1502.26	3	10.86	1458.29	1436.05	0.72	97.07
80	664.798	**655.746**	917.42	15	16.35	853.75	78.91	1.78	93.06
90	604.152	**604.152**	373.52	1	4.20	349.55	349.55	1.12	93.58
100	559.017	**555.662**	123.78	10	6.82	91.64	12.70	5.51	74.04
Average				11	128.21	3266.13	476.39	4.82	86.12

Table 9: Solutions for $n = 1002$ TSP-Lib using the Revised Drezner's Algorithm starting from Best Heuristic Value

[a] This excludes computational time for the heuristic step.

$^+$ could not be computed due to computer memory.

	Best Heuristic	Optimal (or Best) Solution							
p	Z	Z^*	Loop CPU Time (secs)[a]	# Loops	Max Circles (secs)	CPLEX (secs)		Max Circles (%)	CPLEX (%)
						Total	Last Loop		
10^+	2897.490	–	–	–	–	–	–	–	–
20^+	1886.820	–	–	–	–	–	–	–	–
30	1466.970	**1466.970**	29522.00	2	1605.09	26403.90	12725.60	5.43	89.44
40	1236.380	1235.660*	86400.00	5	199.23	86150.77	19277.17$^\perp$	0.23	99.71
50	1060.820	1060.420*	86400.00	2	48.08	85933.90	400.00$^\perp$	0.06	99.46
60	941.870	940.483*	86400.00	7	43.10	86333.90	18895.60$^\perp$	0.05	99.90
70	844.967	**843.801**	13454.40	12	38.72	13323.10	6278.02	0.29	99.02
80	774.764	**774.764**	51229.30	1	9.45	51164.10	51164.10	0.02	99.87
90	720.625	**706.145**	5942.07	33	46.91	5750.88	119.51	0.80	96.78
100	662.936	**658.997**	37388.90	15	20.53	37273.30	6915.90	0.05	99.69
Average				10	251.39	49041.73	14471.99	0.87	97.98

Table 10: Solutions for $n = 1323$ TSP-Lib using the Revised Drezner's Algorithm starting from Best Heuristic Value

[a] This excludes computational time for the heuristic step.
* best feasible solution found within 86400 seconds.
$^+$ could not be computed due to computer memory.
$\perp$ no feasible solution found in the last iteration within the time limit allowed.

It is important to note that for smaller values of p (i.e. $p = 10$ for $pr1002$ and $p \leq 20$ for $rl1323$) computer memory becomes an issue leading to no results being found. This could be due the initial upper bound being higher in these instances, leading to a relatively large number of circles being considered and thus making the ILP model too big to be handled.

In summary, the results show that the revised Drezner optimal algorithm can now find very good and even optimal solutions for these large data sets. In addition, we can also claim that optimal solutions are found for the first time for the large data sets such as $n = 575$, $n = 1002$ and $n = 1323$ and some for $n = 783$ while requiring a reasonable amount of computational time only for such strategic decision problems.

8. Conclusions and Suggestions

This paper has revisited an optimal and interesting algorithm proposed by Drezner (1984) thirty years ago to solve the continuous $p-$centre problem. Opportunities to improve the algorithm were highlighted, and enhancements were developed, mathematically supported and empirically tested. The two areas of interest include the way the $Z-$maximal circles are

identified from one iteration to the next, and the proposed adaptive CPLEX scheme to find a compromise solution at each iteration between the quality of the feasible solution and the optimal solution when solving the covering problem CP_1.

The proposed algorithm was tested on five existing TSP-Library data sets, namely $pr439$, $rat575$, $rat783$, $pr1002$ and $rl1323$ for $p = 10, \ldots, 100$. The results show that the enhanced optimal method gives a very significant decrease in computational time which sometimes reaches an average reduction of 96%, yielding an algorithm that is superior, faster and more efficient meaning that it can be used to optimally solve the continuous $p-$centre problem for large data sets for the first time.

One potential research avenue which we believe to be useful would be to incorporate a fast and good heuristic to generate a feasible solution to the covering problem CP_1 instead of using CPLEX all the time. However, as mentioned earlier, at a certain iteration CPLEX or equivalent commercial solver needs to be used to prove infeasibility as this task is mandatory and cannot be performed by a heuristic to guarantee infeasibility. This leads to adopting a new strategy that could combine the exact method and the heuristic approach to solve CP_1 which would identify the appropriate time when the switching from using the heuristic to CPLEX should take place. This is a challenging but interesting task that deserves a thorough investigation. Lastly, research issues related to the tightening of the checking area and in the way the demand points are recorded during the search could also be worth enhancing even further. These aspects are currently being investigated.

Acknowledgments

The authors would like to thank the referees for their constructive comments that improve both the content and the presentation of the paper. We are also grateful to Dr Abdalla Elshaikh for the $C++$ code of the H_2 heuristic, and Professor Frank Plastria for his insightful comments. The first author would also like to thank EPSRC for her PhD studentship.

References

Brandenberg, R., & Roth, L. (2009). New algorithms for $k-$center and extensions. Journal of Combinatorial Optimization, 18, 376-392.

Caruso, C., Colorni, A., & Aloi, L. (2003). Dominant, an algorithm for the $p-$center problem. European Journal of Operational Research, 149, 53-64.

Chen, R. (1983). Solution of minisum and minimax location-allocation problems with Euclidean distances. Naval Research Logistics Quarterly, 30, 449-459.

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., & 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.

Centre of Logistics and Heuristic Optimisation. (2015). http://www.kent.ac.uk/kbs/research/research-centres/clho/datasets.html, Kent Business School, The University of Kent.

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

Drezner, Z. (1984). The $p-$centre Problem - Heuristic and Optimal Algorithms. Journal of Operational Research Society, 35, 8, 741-748.

Drezner, Z., & Shelah, S. (1987). On the complexity of the Elzinga-Hearn algorithm for the $1-$centre problem. Mathematics of Operations Research, 12 (2), 255-261.

Drezner, Z. (2011). Continuous Center Problems, In H. A. Eiselt & V. Marianov (Eds.) Foundations of Location Analysis (pp. 63-78). New York: Springer.

Elloumi, S., Labbe, M., & Pochet, Y. (2004). A new formulation and resolution method for the $p-$center problem. INFORMS Journal of Computing, 16, 84-94.

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.

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

Hakimi, S. L. (1965). Optimum Location of Switching Centers in a Communications Network and some related Graphical Theoretic Problems. Operations Research, 13, 462-475.

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

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.

Megiddo, N., & Supowit, K. (1984). On the Complexity of some common geometric location problems. Society for Industrial and Applied Mathematics, 13 (1), 182-196.

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.

Plastria, F. (2002). Continuous Covering Location Problems. In Z. Drezner, & H. W. Hamacher (Eds.), Facility Location: Applications and Theory (pp. 37-72). Springer, New York.

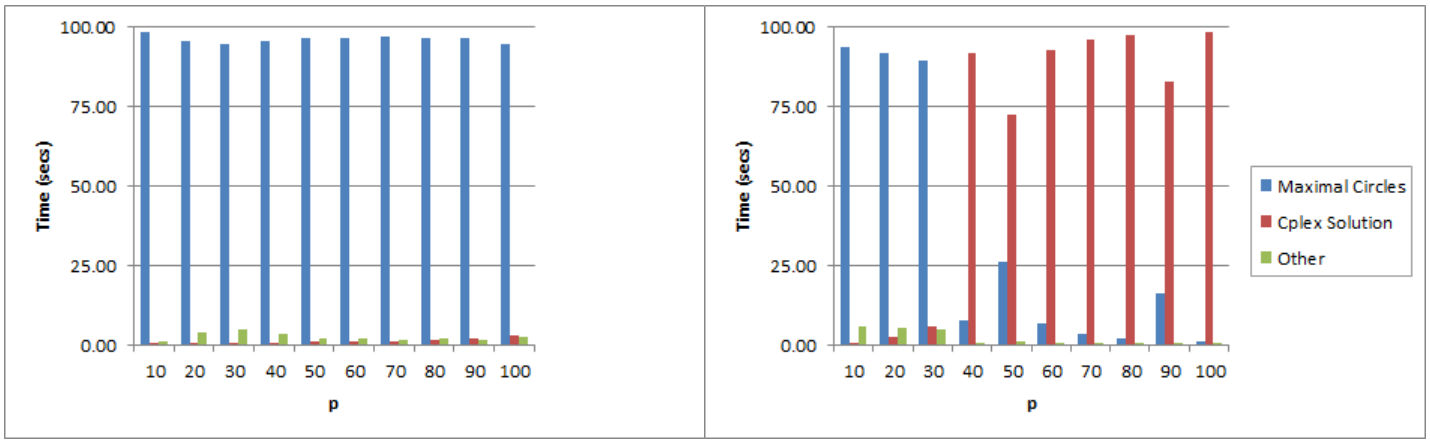
Richard, D., Beguin, H., & Peeters, D. (1990). The location of fire stations in a rural environment: a case study. Environment and Planning, 22, 39-52.

Travelling Salesman Problem Library. (2015), http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

Wei, H., Murray., A. T., & Xiao, N. (2006). Solving the continuous space $p-$center problem: planning application issues. IMA Journal of Management Mathematics, 17 (4), 413-425.

Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). In H.Maurer (Ed), New Results and New Trends in Computer Science (pp.359-370). Springer.

Appendix



(a) $n = 439$ (b) $n = 575$

Figure A.1: Comparing time spent to calculate $Z-$maximal circles, the cplex solution and other

	H_2 Heuristic		Optimal Solution						
p	Z	CPU Time	Z^*	Loop CPU Time (secs)[a]	# Loops	Maxi Circles (secs)	CPLEX (secs)	Maxi Circles (%)	CPLEX (%)
10	1716.510	96.88	1716.510	6252.72	2	6154.93	36.39	98.44	0.58
20	1169.540	170.28	1029.715	56753.00	36	54203.60	297.90	95.51	0.52
30	975.000	205.36	739.193	37017.10	49	35024.50	222.96	94.62	0.60
40	874.271	218.90	580.005	31355.00	67	29986.40	209.61	95.64	0.67
50	580.005	235.61	468.542	4939.25	38	4781.67	59.91	96.81	1.21
60	570.088	246.86	400.195	4956.45	47	4794.88	57.77	96.74	1.17
70	503.271	256.30	357.946	3170.89	46	3076.31	39.04	97.02	1.23
80	467.039	300.01	312.500	2186.27	53	2109.08	37.33	96.47	1.71
90	391.511	276.20	280.903	1258.22	48	1214.45	23.80	96.52	1.89
100	315.486	332.53	256.680	462.30	32	437.38	13.93	94.61	3.01
Average	756.272	233.89	614.218	14835.12	42	14178.32	99.87	96.24	1.26

Table A.2: Optimal Results using the original Drezner's Algorithm $n = 439$ TSP-Lib with the CP_1 formulation at each iteration

[a] This excludes computational time for the H_2 heuristic.

	H_2 Heuristic		Optimal Solution						
p	Z	CPU Time	Z^*	Loop CPU Time (secs) [a]	# Loops	Maxi Circles (secs)	CPLEX (secs)	Maxi Circles (%)	CPLEX (%)
10	69.426	98.34	67.926	83898.60	10	78805.90	351.59	93.93	0.42
20	48.107	175.62	45.475	19087.06	11	17513.80	519.37	91.75	2.72
30	39.655	238.26	35.556	9743.91	14	8698.37	577.51	89.27	5.93
40	33.365	296.90	30.063	41733.00	11	3240.15	38342.30	7.76	91.88
50	30.336	403.76	25.826	9612.61	15	2515.16	6985.51	26.17	72.67
60	27.951	422.18	23.163	28344.00	18	1938.64	26327.70	6.84	92.89
70	25.578	558.86	20.858	40256.90	20	1449.39	38756.30	3.60	96.27
80	24.135	535.90	19.026	40181.70	17	892.371	39247.90	2.22	97.68
90	21.932	743.20	17.460	4260.10	18	696.769	3532.50	16.36	82.92
100	20.402	795.13	16.420	33694.00	15	405.90	33262.20	1.20	98.72
Average	34.089	426.81	30.177	31081.242	14.9	11615.65	18790.29	33.91	64.21

Table A.3: Optimal Results using the original Drezner's Algorithm for $n = 575$ TSP-Lib with the CP_1 formulation at each iteration

[a] This excludes computational time for the H_2 heuristic.