Neighbourhood Reduction in Global and Combinatorial Optimization: The Case of the *p*-Centre Problem

Said Salhi[a*] and Jack Brimberg[b]

[a] Centre for Logistics and Heuristic Optimisation (CLHO), Kent Business School, University of Kent, Canterbury, UK
s.salhi@kent.ac.uk
[b]Department of Mathematics and Computer Science, Royal Military College of Canada, Kingston, ON K7K 7B4, Canada
Jack.Brimberg@rmc.ca

Abstract

Neighbourhood reductions for a class of location problems known as the vertex (or discrete) and planar (or continuous) *p*-centre problems are presented. A brief review of these two forms of the *p*-centre problem is first provided followed by those respective reduction schemes that have shown to be promising. These reduction schemes have the power of transforming optimal or near optimal methods such as metaheuristics or relaxation-based procedures, that were considered relatively slow, into efficient and exciting ones that are now able to find optimal solutions or tight lower/upper bounds for larger instances. Research highlights of neighbourhood reduction for global and combinatorial optimisation problems in general and for related location problems in particular are also given.

Keywords- neighbourhood reduction, *p*-centre problem, continuous and discrete spaces, heuristic search, optimal methods

*Corresponding author

1. Introduction

The objective of the *p*-centre problem is to select *p* sites to locate new facilities in order to minimise the maximum distance or travel time between a set of demand points and the facilities closest to them. This problem, originally formulated on a graph by Hakimi (1964), is usually categorised as either the vertex *p*-centre problem or the absolute (or planar) *p*-centre problem. In the former, which is the discrete case, the optimal facilities are selected from a given set of potential sites (vertices) which can be either the demand points or other known sites. However, in the latter case the facilities can be located anywhere in the plane.

An illustrative example for the vertex 1-centre problem and its counterpart the planar 1-centre is given in Figure 1, where four fixed points (or vertices) are located at (0,0), (1,0), (0,1) and (1,2). Here the optimal solution locates the single facility at (0, 1) for the discrete case with resulting minimum objective function value of $\sqrt{2}$, the Euclidean distance from (0,1) to (1,2) or (1,0). Meanwhile the optimal planar location is at (0.5,1) situated half way between the two points (0,0) and (1,2) that are furthest apart with objective function value now reduced to $\frac{\sqrt{5}}{2}$.

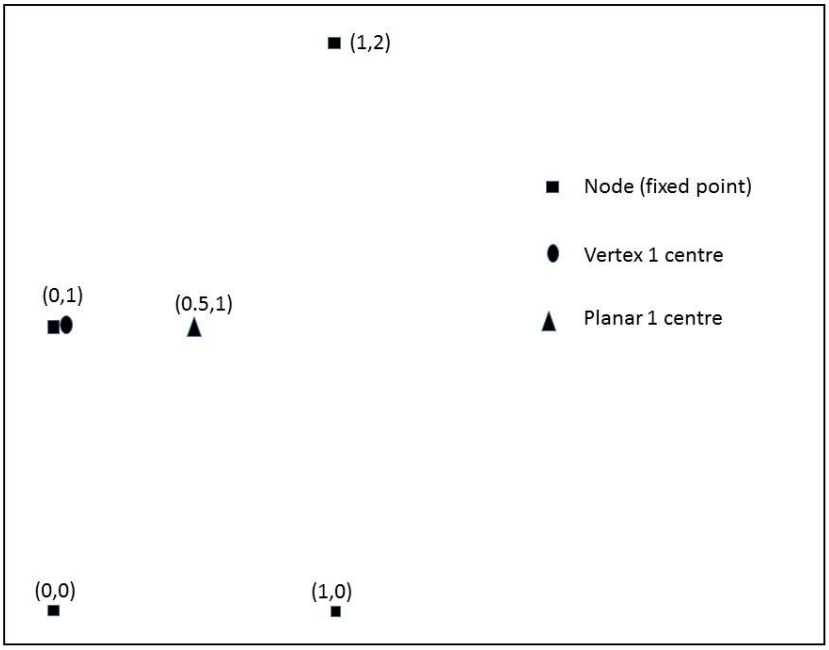

Figure 1: A vertex and planar 1-centre problem with 4 fixed points.

When the aim is to locate facilities on a network, and the possible sites are restricted to the vertices of the network (i.e., the discrete case), the problem is known as the general p-centre problem, whereas if the facilities are also allowed to be located at any points on the edges of the network, the problem is then referred to as the general absolute p-centre problem. For instance, for the case of one centre, in the former the idea is to choose the vertex that has the smallest distance to its furthest vertex, whereas for the latter the idea is to find a point along an edge that leads to the smallest distance to the furthest vertex. This latter class of location problems is not discussed here, but more details including illustrative examples can be found in Eiselt and Sandblom (2004). For a general discussion and analysis on continuous location problems, the reader will find the chapter by Drezner (2011) to be interesting and informative.

Heuristic search embodies the field of practical optimization for solving combinatorial (discrete) and global (continuous) optimization problems. The word 'heuristic' derives from a Greek word meaning 'I discover or find'. In brief, it can be considered as a combination of mathematical logic, statistical ideas, computer science experience and insight of the problem. Heuristic search aims to provide good quality solutions in a reasonable amount of computing time while not necessarily guaranteeing an optimal solution. Heuristics are used when exact methods fail due to either entrapment at local optima or excessive use of memory or computing time. Heuristic search can be classified under several categories. For instance, Salhi (2017) provided the following one**:** improving only heuristics, not necessarily-improving heuristics, population-based heuristics and hybridization. To enhance the efficiency of these approaches, neighbourhood reductions and data structures are usually constructed and embedded into the search. The first one attempts to avoid unnecessary calculations whereas the latter attempts to store information that can be used in subsequent iterations. In this chapter we concentrate on the former.

Neighbourhood reduction aims to eliminate moves or checks that cannot lead to an optimal solution for the case of exact methods, or are unlikely to affect the global best solution in the case of heuristics. It is challenging to design powerful neighbourhood reduction schemes that can cut computing time as much as possible without (or slightly) affecting the quality of the solution. Determining a good balance between the depth of the cut and the retention of solution quality at a reasonably high level is a challenging issue. Having good insight into the structure of the problem is useful as this helps in defining and

designing the right neighbourhood reduction scheme. This mechanism, also known as reduction test, can be either dynamic which adapts as the search progresses, or deterministic which is usually defined from the outset. For more information on this research design issue, see Salhi (2017).

This chapter is organized into three parts:

(i) a brief review of both vertex and planar *p*-centre problems with a focus on the contributions by Drezner;

(ii) a review of some neighbourhood reductions that are observed to be very useful for solving the vertex and planar p- centre problems; and

(iii) a discussion of some key research items on neighbourhood reduction that could be worth exploring.

2. The p-centre problem

We organise this section into two subsections. The first deals with the (discrete) vertex *p*-centre problem, while the second discusses the (continuous) planar *p*-centre problem.

2.1 The vertex *p*-centre problem

The vertex *p*-centre problem, also known as the multi-facility minimax location problem, aims to optimally locate *p* facilities among a finite number of potential sites and to assign demand points to these open facilities in order to minimise the maximum distance between demand points and their nearest facility. There are two main formulations, as a binary linear program (BLP), and as a set covering problem (SCP).

2.1.1 *The BLP formulation*

Let

(I, J) : the set of demand points (or customers) $(i \in I = \{1,...,n\})$ and set of potential facility sites $(j \in J = \{1,...,m\})$,

$d(i, j)$: the distance between customer *i* and potential site *j* (Euclidian distance is used in our study);

p : the required number of facilities;

$$Y_{ij} = \begin{cases} 1 \text{ if customer } i \text{ is assigned to a facility at location } j \\ 0 \text{ otherwise} \end{cases}$$

$$X_j = \begin{cases} 1 \text{ if a facility is opened at location } j \\ 0 \text{ otherwise} \end{cases}$$

Z : the maximum distance between the customers and their closest facilities.

The problem (BLP) is then formulated as follows:

Minimize Z (1)

Subject to

$$\sum_{j \in J} Y_{ij} = 1 \quad \forall i \in I \tag{2}$$

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

$$Y_{ij} - X_j \leq 0 \quad \forall i \in I, j \in J \tag{4}$$

$$Z \geq \sum_{j \in J} d(i, j) Y_{ij} \quad \forall i \in I \tag{5}$$

$$X_j \in \{0,1\} \quad \forall j \in J \tag{6}$$

$$Y_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \tag{7}$$

The objective (1) refers to the minimization of the maximum distance between a customer and its nearest facility. Constraints (2) guarantee that each customer is assigned to exactly one facility; constraint (3) limits the number of open facilities to be *p*; while constraints (4) ensure that a customer can only be allocated to an open facility. Constraints (5) define the maximum distance between a customer and its closest facility. Constraints (6) and (7) refer to the binary type of the decision variables. Note that the binary type constraints on the Y_{ij} in (7) can be replaced by non-negativity constraints without affecting the optimal solution, since the minimization objective will force customers to be assigned to their nearest facilities.

The *p*-centre problem is known to be NP-hard (Kariv and Hakimi, 1979). Thus it follows that only small to medium size instances of this problem can be solved optimally using commercial optimization software such as CPLEX, LINDO, GUROBI or Xpress-MP, and it becomes more difficult to tackle for relatively large instances. One idea is to aggregate customers leading to a smaller problem which is more manageable. However, it is worth noting that such an aggregation-based approach, if not considered carefully, could lead to

poorer quality solutions due to the loss of information. Another approach is to address the problem in its entirety by adopting powerful metaheuristics or mat-heuristics. For instance, Irawan, Salhi and Drezner (2016) develop a powerful hybridisation of VNS and ILP formulations by embedding intelligent neighbourhood reduction schemes.

2.1.2. A set covering-based model

The minimax problem can also be solved optimally using a Set Covering Problem (SCP)-based approach. Given a covering distance (or response time) D, SCP aims to find the minimum number of facilities and their locations so that each customer is served by a facility within D distance from it.

Let

$$a_{ij} = \begin{cases} 1 \text{ if customer } i \in I \text{ can be covered by a facility sited at } j \in J \text{ (i.e., } d(i, j) \le D) \\ 0 \text{ otherwise} \end{cases}$$

The SCP can be formulated as follows:

Minimize $$\sum_{j \in J} X_j \tag{8}$$

Subject to

$$\sum_{j \in J} a_{ij} X_j \ge 1 \quad \forall i \in I \tag{9}$$

$$X_j \in \{0,1\} \quad \forall j \in J \tag{10}$$

The objective (8) is to minimise the number of opened facilities. Constraints (9) guarantee that each customer is covered by at least one facility located within the threshold D and constraints (10) refer to the binary variables.

The minimax problem is optimally tackled by recursively solving a sequence of SCPs for given values of D using a binary search. For instance, Daskin (2000) adopted this approach, initially presented by Minieka (1970), on a general graph with all edge distances restricted to integers. Efficient exact algorithms for solving the vertex *p*-centre problem include, for example, Ilhan and Pinar (2001), Elloumi, Labbe and Pochet (2004), Al-Khedairi and Salhi (2005), Salhi and Al-Khedairi (2010), and Irawan, Salhi and Drezner (2016). The latter ones incorporate neighbourhood reductions which are discussed in Subsection 3.2.

2.2 The planar *p*-centre problem

Continuous location problems are about generating sites for one or more facilities in the plane. Though the obtained solutions may not be feasible as some facilities may end up in the middle of a city or a lake, they can still be used as greenfield solutions (ideal solutions). Given that the continuous problem can be a good approximation of its discrete counterpart especially when the network has a large number of potential sites, getting the ideal solution could provide valuable information for decision makers when selecting the final sites. As the data gathering task for a large network can be very expensive to conduct, the continuous model may also be used to reduce the number of potential sites to a few promising ones, thus making the problem more manageable.

From a theoretical view point, the continuous *p*-centre problem is also interesting as it has a geometrical interpretation. For example, the single unweighted facility location problem (i.e., $p = 1$) reduces to finding the smallest circle that encloses all the customers, with the centre of the circle being the location of the new facility. In a similar way, the continuous *p*-centre problem with $p > 1$ may be interpreted as finding the centres of p circles that encompass all the customers where the radius of the largest circle is made as small as possible.

The (weighted) *p*-centre problem can also be described as a MinMaxMin type problem with formulation given by Drezner (1984*a*):

$$Min\, Z(X) = \underset{i=1,...,n}{Max}[w_i \underset{j=1,...,p}{Min}\, d(P_i, X_j)]$$

where the additional notation is defined as:

$P_i = (a_i, b_i)$: the given location of demand point i $(i = 1,...., n)$

$w_i > 0$: the weight of demand point i $(i = 1,...., n)$

$X_j = (x_j, y_j)$: the unknown location of new facility j with $X_j \in \Re^2; j = 1,..., p$

$X = (X_1,...., X_p)$: the vector of decision variables containing these p facility locations

$d(P_i, X_j)$: the Euclidean distance between P_i and X_j $(i = 1,...., n; j = 1,...., p)$

Note that the *unweighted* model is normally considered as in the discrete *p*-centre problem given above with equal weights $(w_i = 1,\ i = 1,..., n)$.

The single facility minimax location problem (1-centre) in continuous space has a long history. The English mathematician James Joseph Sylvester (1814-1897) first posed the problem in 1857, and then a few years later, in 1860, put forward an algorithm to solve it. The problem was dormant for over a century until Elzinga and Hearn (1972) presented an optimal geometrical-based algorithm that runs in polynomial time. Since then, other authors attempted some speed up procedures, such as Hearn and Vijal (2002), Xu, Freund and Sun (2003), and Elshaikh, Salhi and Nagy (2015), and references therein. Some of these enhancements use simple but effective reduction schemes, which are discussed in Subsection 4.1. For an informative review including the history of this problem, see Drezner (2011) and references therein .

There is, however, a relative shortage of studies dealing with the problem for larger values of *p* (see Plastria (2002), and Callaghan, Salhi and Brimberg (2018)). Chen (1983) is among the first to tackle the *p*-centre problem in the plane. The problem is shown to be NP-hard in Megiddo & Supowit (1984). For a fixed value of *p*, the problem can be solved in polynomial time $O(n^2 p)$ as shown by Drezner (1984a), though it requires an excessive amount of computational effort for larger values of *n* and *p*. Due to the non-convexity of the objective which is a function of the location variables, this problem also falls in the realm of "global optimization".

For the case of the 2-centre problem with Euclidean distances in the plane $(i.e., p = 2)$, there is an interesting optimal algorithm by Drezner (1984b). The idea is that the entire customer set can be split into two separate sub-problems by a straight line, where each can be optimally solved as a 1-centre problem. However, as there $\frac{n(n-1)}{2}$ possibilities, the problem becomes difficult though still polynomial. The method can be extended to larger *p*, where more than one line would be needed, but this problem becomes much more difficult. A scheme on how to proceed from one set of *p* lines (*p* clusters) to another is an exciting exercise that could be worth exploring.

Constructive heuristics were the first to emerge for larger values of *p*. These use the iterative locate-allocate procedure initially proposed by Cooper (1964) for the Weber problem for local improvement, and are based on the commonly used *add*, *drop* and *swap* moves. For instance, Drezner (1984a), and Eiselt and Charlesworth (1986) were among the first to develop such methods.

Drezner (1984a) also devised a nice optimal algorithm using the idea of Z-maximal circles. For a given Z, all maximal circles are defined and either a corresponding set covering problem or a feasibility problem is solved. Starting from a lower bound for Z, successive problems are solved until a feasible solution is found (i.e., the solution has p circles with each customer being encompassed by at least one circle). This optimal algorithm was originally very slow but has since developed into a fast and powerful approach that can solve large instances to optimality. This is achieved by incorporating suitable neighbourhood reductions into the search which are discussed in Subsection 4.3.1.

Several years later, Chen and Chen (2009) developed a relaxation method based on Chen and Handler (1987) to optimally solve the problem. The idea is to start by solving a reduced problem containing a subset of demand points from the original problem and then gradually adding some points to the current subset until the optimal solution is feasible for the original problem. This interesting relaxation-based approach is also revisited and its efficiency much enhanced in Subsection 4.3.2.

3 Neighbourhood reduction for the vertex p-centre problem

We first present basic reduction schemes which can be embedded efficiently into the brute force approach, also known as the complete enumeration technique. These are followed by those more advanced neighbourhood reductions that are adopted primarily for optimal methods, metaheuristics and mat-heuristics for the case of the p-centre problem.

3.1 Brute force approach

The idea is to evaluate all combinations of p possible facility sites out of the n potential sites. For each combination allocate each customer to its nearest facility, leading to p clusters, and choose the one that yields the maximum distance from the allocated customers to the centre (their nearest facility). The optimal solution is one with the minimum of these maxima. This complete enumeration technique (CET), though naive, can be used to guarantee the optimal solution for small values of p (< 5), even when n is around 100, without the use of commercial optimisers or even the use of any heuristic. This simple and rudimentary

approach if applied blindly will evaluate all the $C_p^n = \frac{n!}{p!(n-p)!}$ combinations, and will fail rapidly when $p \geq 5$ even for $n = 100$.

However, with simple reduction rules, we can improve its efficiency drastically still without recourse to advanced methods. The following four rules, which are given in Al-Harbi (2010), are briefly discussed here.

(a) CET is coded in *p* nested loops (in an ordered fashion) leading to any two successive configurations being different by one facility only. In the allocation process, for a given customer i we have two options whether this customer has lost its initial facility and hence needs to be checked against all *p* facilities including the new one; otherwise one comparison between its original assignment and the new one is evaluated. For example, the instance *(Pmed1)* from the ORLIB with $n = 100$ and $p = 5$ required 506 secs, whereas with this simple reduction it needed only 348 secs. The CET and simple reduction were coded in C++ and performed on a PC i7 with 1.5GHz processor and 512 MB of RAM. However, for the instance *Pmed6* $(n = 200; p = 5)$, both versions were unable to obtain the optimal solution after 5 hours of CPU time. Here, the blind approach exploited 42% of the total number of combinations whereas the enhanced one used 70% instead, leading to better chances of obtaining an improved solution, though optimality cannot obviously be guaranteed.

(b) Besides (a), we can also record the second closest facility for each customer. Though this adds extra computational storage, it reduces the overall computing time. Using the same example with $n = 100$ and $p = 5$, this simple recording task enables the optimal solution to be found within 321 secs. For $n = 200$ and p=5, though the optimal solution is still not guaranteed, 77% of combinations were now evaluated.

(c) Note that both operators (*(a)* and *(b)*) are not only applicable to this problem but are commonly used in many other combinatorial problems where an assignment is required. We take into account additional insights unique to the *p*-centre problem. Given the objective is to minimize the maximum coverage, it is clear that once we have one feasible solution with a value of Z, this can be used to terminate the evaluation of a given configuration if one customer happens to have its distance to its nearest facility larger than Z. In other words, there is no need to continue checking the other customers for this particular configuration. The upper bound *Z* can be updated as the search proceeds. This reduction scheme systematically leads to rejecting several inferior configurations early on leading to a massive

reduction in the overall computing time. This basic rejection scheme obtains the optimal solutions for both instances (*Pmed1* and *Pmed6*) within the maximum 5 hours allowed (18000 secs), requiring only 54 secs and 11546 secs, respectively.

(d) This is an extension of (c) where for each customer $i = 1,...,n$, a set of facilities $F_i = \{ j = 1,...,n \mid (d(i, j) \le Z \}$ is constructed (usually updated as the search goes on). If a given configuration does not contain at least one facility in F_i , there is no need to continue the allocation of other customers as this configuration is inferior. This dynamic reduction rule, which is based on the current Z , renders the brute force even faster by obtaining the optimal solutions for *Pmed1* and *Pmed6* in 47 secs and 2634 secs only, respectively. This rule dominates the one that also states that for a given customer, a facility configuration that includes one of its furthest $(p-1)$ facilities is systematically inferior and hence needs to be discarded.

The above rules demonstrate that the information in a given problem may be used to eliminate several redundant computations if appropriate neighbourhood reductions are designed. However, even with such elimination rules, the brute force approach is still limited to smaller values of *n* and *p*. Having said that, the effective use of neighbourhood reduction is still able to reduce by as much as some 90% the time attributed to unnecessary computations. The impact is even more significant when these reduction schemes are embedded within powerful meta-heuristics or optimal algorithms as will be shown in the rest of this chapter.

3.2 Set covering-based approach

The approach using SCP, as given by Daskin (2000), is shown in Figure 2. In this section, we revisit some of its steps to enhance its efficiency.

Step 1- Set $L = 0$ and $U = \underset{i,j}{Max}\, d(P_i, P_j)$.

Step 2- Compute the coverage distance $D = \dfrac{L+U}{2}$

Step 3- Solve the Set Covering Problem (SCP) using D as the covering distance and let v be the optimal number of facilities obtained.

Step 4- If $v \le p$ (i.e., the solution is feasible for the *p*-centre problem), set $U = D$;
else (ie., $v > p$, the solution is infeasible) set $L = D$.

Step 5- If $U - L \le 1$, record U as the optimal solution and stop, otherwise go to Step 2.

Figure 2: The basic SCP algorithm

Revisiting Steps 1 and 5

For instance, Al-Khedhairi and Salhi (2005) proposed some basic changes in Step 1 when initialising the bounds L and U by re-defining

$L = \underset{j}{Max}\, \underset{i}{Min}\, d(P_i, P_j)$ and $U = \underset{i}{Min}\, \underset{j}{Max}\, d(P_i, P_j)$.

Also, to guarantee that the elements of the distance matrix in Step 2 are used only, D is redefined slightly by setting $D = G(\dfrac{L+U}{2})$ where $G(x)$ represents the nearest value to x in the distance matrix.

In addition, to terminate the search as early as possible in Step 5 and avoid redundant checks, the set $S = \{d(P_i, P_j) : L < d(P_i, P_j) < U\}$ is introduced. If $S = \varnothing$ (ie., there are no distance values between L and U), the search terminates even if $U - L > 1$, with the optimal solution being U. In addition, if |S|=1, there is one element left to assess only, say D and go to step 3. The optimal solution either remains at U or D if the new SCP solution happens to be feasible. These schemes are tested on all instances of the OR-Lib ($n = 100$ to 900, and $p = 5$ to 90) where a 15% average reduction in the number of SCP calls is obtained. For the TSP-Lib data set ($n = 1060$ and $p = 10, 20, ..., 150$), a more significant average reduction of over 28% is recorded.

Further tightening of U and L in Step 1

In step 1, a simple tightening of U can be found easily just by running a multi-start approach and choosing $U = Min(\underset{i}{Min}\, \underset{j}{Max}\, d(P_i, P_j), Z_H)$ with Z_H being the best solution of all the runs. For instance, when $Max(n, 500)$ multi starts are adopted, the above results

for the OR-Lib and TSP-Lib improved even more leading to an average reduction in the number of iterations from the original implementation of 23% and 41% respectively.

A further tightening of U and L can be obtained by using the solution from a powerful metaheuristic Z_H as U leading to a potential lower bound $L = \alpha U$ with $\alpha = 0.7$ or 0.8. The tighter Z_H is, the closer α is to 1. The power of this setting is that even if αU fails to be a `true' lower bound, its value will automatically become the upper bound instead, and the lower bound is recomputed again as $L = \alpha U$ with α kept the same or slightly reduced until a range (L,U) is derived. It is worth noting that there are no redundant calculations when assessing the SCP for L. Salhi and Al-Khedhairi (2010) proposed this implementation with interesting results using a multi-level heuristic originally proposed by Salhi and Sari (1997). Recently, Irawan, Salhi and Drezner (2016) adopted the above methodology with two distinct changes. The upper bound in step 1 is obtained by VNS instead (see Hansen et al, 2010), and also an ordered list of the distance matrix is constructed to easily identify the elements in the set S. As the upper bound produced by VNS in Step 1 may be close to the optimal solution, α can be set in the range [0.8-0.9] leading to an even tighter $L = \alpha U$. Note that the value of L must also exist in the distance matrix, and hence, is set to the nearest such value to αU.

Other tightening of L

An interesting two phase approach which shares some similarities with the SCP is based on solving the feasibility of the following covering problem (CP) instead (Ilhan and Pinar, 2001). Note that the notation remains as given in Subsection 2.1.

$$\{ \quad \sum_{j \in J} a_{ij} X_j \geq 1 \quad \forall i \in I \; ; \; \sum_{j \in J} X_j \leq p \; ; \; X_j \in \{0,1\} \quad \forall j \in J \quad \}$$

The idea is that if a relaxed CP (i.e., $0 \leq X_j \leq 1 \quad \forall j \in J$) which is much quicker to solve does not provide a feasible solution for a given D, there is no need to solve the integer problem. This phase one is similar to SCP in Figure 2, except that Step 3 is based on solving the relaxed CP. Once a feasible solution is obtained in phase one, phase two is activated where CP is solved with the corresponding D. If the integer problem is not feasible, then a tight lower bound $L = D$ can be used. Note that phase two in Ilhan and Pinar (2001) does not follow the SCP algorithm given in Figure 2 but attempts to solve the integer CP by gradually

increasing D to the next minimum in the distance matrix until an integer solution of CP is found.

3.3 Variable neighbourhood search based reduction

Irawan, Salhi and Drezner (2016) introduce some elimination rules to avoid computing unnecessary moves when adopting variable neighbourhood search (VNS) for solving the discrete location problem. In brief, VNS is a metaheuristic originally developed by Mladenovic and Hansen (1997) for tackling combinatorial and global optimisation problems. VNS attempts to avoid local optimality by systematically changing neighbourhoods in a shake (or perturbation) operation, usually starting from the smallest (easiest to compute) to the largest one. The idea is to start with an initial local solution, and then generate a random point in its first neighbourhood (a shake). A local search is then applied from this neighbouring point. If the new local solution is an improvement, a move is made to it, and the search reverts back to the first neighbourhood; otherwise a random point in the next neighbourhood (usually a larger one) of the current solution is generated (the neighbourhood change step), and the process repeats. The cycle of shaking and local search from the smallest to largest neighbourhood or to the next improvement, whichever comes first, is repeated until typically a specified limit on execution time is reached. Salhi (2017) classified metaheuristics into four classes, namely, improvement-only heuristics (composite heuristics, multi-level heuristics, GRASP, perturbation methods, large neighbourhood search, iterated local search, guided local search, etc), not necessarily improving heuristics (simulated annealing, tabu search, threshold accepting, etc), population based (genetic algorithm, scatter search, particle swarm, ant colony, bee algorithms, etc) and hybridisation (between heuristics, or between heuristics and exact methods). Basic VNS generates a sequence of improved local solutions and hence falls into the first category above.

Irawan, Salhi and Drezner (2016) speeded up the VNS implementation in both the shaking process and the local search by avoiding the evaluation of non promising calculations. The local search, which is a vertex substitution heuristic, implements a swap move by closing an open facility and replacing it with a closed one. However, instead of removing a facility randomly from the current facility configuration, the facility (say facility j) whose radius (the distance between a facility and its furthest customer) is the largest, say D, is chosen.

Let

$E_j = \{i = 1,...,n \mid d(i,j) \le D\}$: be the set of customers served by facility j and

$\tilde{C}_j = Arg\ \underset{i \in E_j}{Max}(d(i,j))$: be the customer whose distance is D from facility j,

$V_j = E_j \cap \{i = 1,....,n \mid d(\tilde{C}_j, i) \le D \wedge d(\tilde{C}_j, i) \ge \frac{D}{2}\}$: be the subset of potential sites from which to randomly choose an open facility to replace facility j.

In this case, the location of the new open facility is restricted so as not to be too close to customer $\tilde{C}_j$. This concept of using forbidden regions is shown to be effective when solving the multi-source Weber problem (Gamal and Salhi, 2001) and its capacitated version (Luis et al., 2009). Here, the threshold distance is set to $\frac{D}{2}$ though this can vary; see Figure 3 which is adapted from Irawan, Salhi and Drezner(2016). This reduction scheme considers a fraction of the customer sites only, which can in turn reduce the search space considerably and lead to a substantial cut on the computing time. This can be significant given the local search in VNS is applied a large number of times.

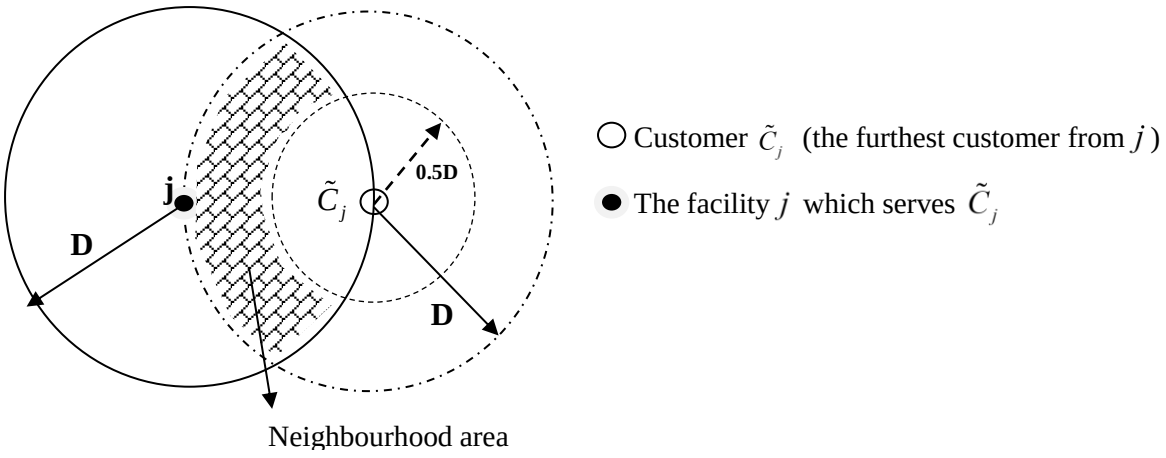

Figure 3: The restricted but guided neighbourhood within VNS

This approach, when integrated as part of a matheuristic, shows to be effective at tackling very large instances up to $n = 71,000$ and $p = 25, 50, 75$ and 100 for both the conditional and unconditional vertex p-centre problems, see Irawan, Salhi and Drezner (2016).

4 Reduction schemes for the planar *p*-centre problem

The planar *p*-centre problem has received relatively less attention compared to its counterpart the vertex p-centre problem especially when it comes to optimal methods. In this section, we first explore elimination techniques for the 1-centre problem though this can be solved optimally in polynomial time. We then present speed-up procedures for the commonly used metaheuristics followed by those devoted to the optimal and relaxation-based methods.

4.1 The 1-centre problem

The problem of determining the optimal location in continuous space for a given cluster of customers turns into finding the centre of the smallest circle that encompasses all the customers. As a circle can be identified by one, two or three critical points only, Elzinga and Hearn (1972) used this concept to develop an $O(n^2)$ geometrical-based optimal algorithm to solve the problem. The idea is to start with two demand points chosen randomly and find the corresponding optimal centre and the radius of the circle. If all demand points are encompassed by the circle, the search terminates; otherwise another point is added and a new circle with the optimal centre for the three points is constructed. If the circle does not cover all demand points, a new uncovered point is added again while one of the three points that becomes redundant is removed. This process continues until a circle that covers all points is found. There are a few studies that looked at this issue. Recently, Elshaikh, Salhi and Nagy (2015) introduced two simple but effective reduction rules into the Elzinga and Hearn (EH) algorithm.

(a) In the starting step, instead of choosing any two points, the four points that make up the smallest rectangle that covers all the points are first identified. For each pair, the corresponding circle is found and the largest one is chosen with its critical points as the starting solution.

(b) When selecting the new point to add to the already existing circle with centre X and radius Z, instead of choosing the new point randomly, the one with the largest (weighted) distance to the current centre is selected instead, i.e., $P^* = \text{Arg} \underset{i=1,\ldots,n}{Max}(\text{w}_i\text{d}(\text{P}_i,\text{X})>\text{Z})$.

When tested on a set of randomly generated instances in a square $(0,100)x(0,100)$ with $n = 100$ to 1000, the enhanced EH algorithm required about 30% and 20% of the computing time of the original version for the unweighted and the weighted cases, respectively. More details on the experiments including other less promising rules can be found in Elshaikh (2014). One may assume that there is no need to speed up such a quick polynomial procedure. This will be obviously true if the aim is to solve the problem once or a small number of times only. However, in the *p*-centre problem, this task embedded within a metaheuristic or an optimal method will be called upon several times, and therefore, in our view the enhancement is quite worthwhile. For example, to demonstrate the benefit of these two reduction tests, Elshaikh, Salhi and Nagy (2015) perform an extensive experiment using a simple multi-start with 100 iterations on the $n = 1002$ *TSPLib* instance with $p = 5$ to 25 in increments of 5. It is found that over 32% less updating within EH is required leading to a reduction of over 25% in computational time.

4.2 Reduction within Heuristic-based approaches

We discuss the recent neighbourhood strategies that have proved to be promising when embedded within the powerful metaheuristics used for the *p*-centre problem.

4.2.1 VNS-based heuristics

In the shaking process within VNS, a certain number ($K_{\max}$) of neighbourhood structures is defined $(N_k(X); k = 1,...,K_{\max})$. For the *p*-centre problem, these can be either (a) customer based or (b) facility based.

In (a), $N_k(X)$ can be defined by reallocating k demand points from their original facilities to other ones either randomly or following a certain strategy. Due to the characteristics of the planar *p*-centre problem, the number of critical points that define the largest circle obtains $K_{\max} \leq 3$. The re-allocation of one of these points will in most cases reduce the radius of the largest circle (except in the case of ties). Note that other circles may increase their radii after this allocate-locate procedure, but the solution is accepted as long as Z is reduced.

In (b), $N_k(X)$ is defined by relocating k facilities $(k = 1,..., K_{\max} = p)$ from the current solution. Instead of randomly removing k facilities, the following removal scheme that guides the search more effectively is performed. As the problem is linked to the largest circle and its surrounding circles, it is therefore important to take into account these characteristics when designing the neighbourhood reductions. Two aspects are worth considering here. The first one is connected to the largest circle and the second is linked to those facilities deemed non-promising. For simplicity, consider the largest circle as C_1 and let

CT_k: the centre of circle $C_k; k = 1,..., p$

$\delta_s = d(CT_1, CT_s); s = 1,..., p$: the distance between the centre of C_1 and the centre of C_s

$\gamma(k) = Arg \underset{\substack{s=1,...,p \\ s \neq \gamma(l), l=1,...k-1}}{Min} \delta_s$ and

$C_{\gamma(k)}$: the k^{th} nearest circle to C_1 with $\gamma(1) = 1$ referring to C_1.

The process starts removing the facility at CT_1 (i.e., circle C_1) and assigning it somewhere else as will be discussed shortly. If after a local search the solution is not improved, both facilities located at CT_1 and $CT_{\gamma(2)}$ (i.e., both circles C_1 and $C_{\gamma(2)}$) are then removed, and the process continues until all facilities (i.e., all circles $C_{\gamma(1)}, C_{\gamma(2)}, ...C_{\gamma(p)}$) are removed if necessary. If the solution is improved, the information is updated (i.e., $CT_k, \gamma(k), k = 1,..., p$) and as in VNS, we revert back to the removal of the facility at CT_1 again.

The second aspect is based on identifying those non-promising facilities for removal**.** In our case a facility is considered as non-promising if it encompasses its critical points only (i.e., there are no interior points within the circle). This identification will lead to a saving of $q(q << p)$ facilities which can then be added around the critical points of the largest circle one at a time. For example, Elshaikh, Salhi and Nagy (2015) conducted an experiment for *TSPLib* with *n* = 439 and *p* = 10, 20,…, 100, where a 9% average improvement was obtained. In particular, for the case of $p = 100$, a 34% improvement was observed with 7 facilities being identified as non-promising. Most of the improvements were found with $p \geq 50$.

In both cases, the re-assignment of facilities is also performed using the characteristics of the *p*-centre problem. Instead of inserting a removed facility either randomly anywhere in the plane or at fixed customer sites, the following attractor scheme for insertion is adopted. Using

the largest circle again with radius Z, for each of its critical points (generally 3 or fewer), it can be shown that there is no other facility within distance Z of any of these critical points. This observation is important as it shows that at least one new facility has to be located somewhere inside this region. This statement can be shown empirically but it is also mathematically proven in Mladenovic, Labbe and Hansen (2003). It is therefore important to consider these areas where the new locations will be sited. Within VNS, each time a facility is removed and relocated, the local search is adopted. If the solution is improved, the following updating takes place by defining the largest circle, the neighbourhood for removal and the new area where to locate. For example, when the q non promising facilities are saved, one facility at a time is located randomly in those areas near the critical points of the largest circle, the reallocation process is then used based on avoiding unnecessary repetition of computations. That is, only affected circles have their centres and radii recalculated using their earlier respective centres and radii as the starting solution. The allocation of customers is also performed effectively by considering whether or not a customer lost its original facility. The process is repeated until there are no non-promising facilities remaining or the solution cannot improve anymore. This enhancement, when tested for the $n = 439 TSPLib$ data set, shows a significant improvement. over its original implementation.

The effect of this neighbourhood reduction has also helped to identify adaptively the best value of $K_{\max}$ as well as the best neighbourhood structures $N_k(X)$ that are worth examining. A VNS-based heuristic with all the above ingredients was able to obtain for the first time optimal solutions for larger TSP-Lib instances, see Elshaikh, Salhi and Nagy (2015) for more details.

4.2.2 Perturbation-based metaheuristics

The idea is to perturb the current feasible solution by allowing it to have up to q facilities over or below p. This up and down shifting, which is repeated several times, has the tendency to retain those very promising facilities in the defined set. Salhi (1997) originally put forward this approach for the p median problem which is now successively adopted and extended for the p-centre problem (Elshaikh *et a*l. (2016). The operators 'add', 'drop' and 'swap' are adopted here. The first operator applies when the solution has $p-q$ or p facilities to reach p or $p+q$ facilities, whereas the second operator is used when the solution has

p or $p+q$ to reach $p-q$ or p facilities. The last operator is activated when the solution has exactly p facilities. The 'add' and 'drop' moves are implemented based on the following neighbourhood reductions.

We define the k^{th} covering circle $CC_k = \{P = (x, y) \in \Re^2 \mid d(P, CT_1) \leq \delta_k\}, k = 1, ..., q$ with k facilities being either removed (the 'drop' move) from CC_k or added (the 'add' move) to CC_k. Both the removal and the addition mechanisms are performed using either $k = q$ in one go or gradually adding or removing one facility at a time until it reaches either $p+q, p$ or $p-q$, see Figure 4 for an illustration. Note that the value of q can be made dynamic using some form of learning. Besides, this value does not have to remain the same when the search goes over or under the value of p (i.e., $p-q_1; p+q_2$).

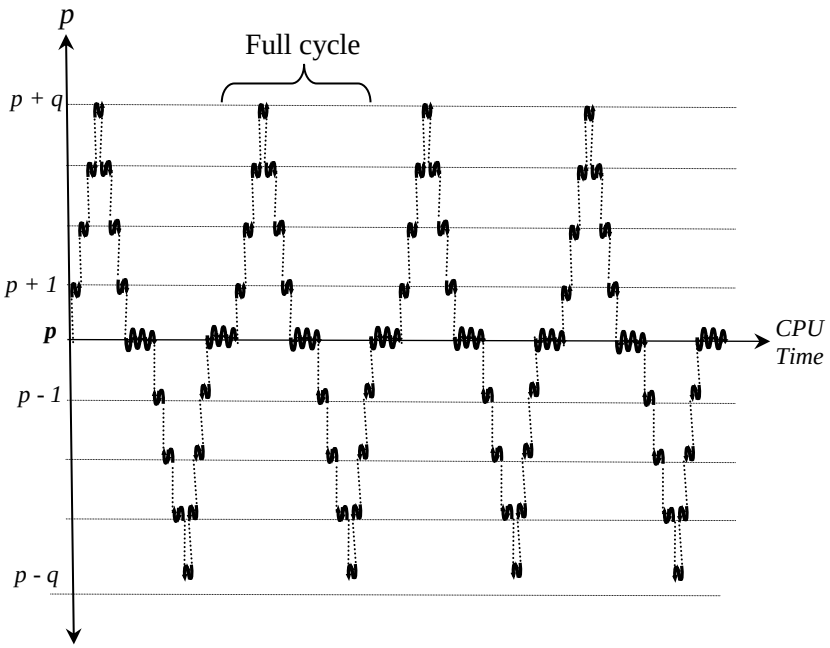

Figure 4: A perturbation-based metaheuristic

In the swap move, a facility in CC_1 is removed and relocated randomly in the continuous space also in CC_1 where a local search is then activated. Note that the Elzinga-Hearn algorithm or its equivalent (see Subsection 4.1) is applied at each solution to obtain the optimal centre for each cluster irrespective whether or not the solution is feasible in terms of the number of facilities. These guided schemes reduce the computing time considerably enabling large *TSPLib* instances with over 1300 nodes and $p = 10, ..., 100$ to be solved efficiently; see Elshaikh *et al.* (2016) for these encouraging results.

4.2.3 Guiding the Search through Forbidden Regions

In the local search, extra care is needed when locating the removed facilities in the continuous space. Here, a new location chosen randomly on the plane may end up by chance being too close to one of the already selected facilities. Given the search is on the plane, one way forward is to construct a small area around each of the existing locations and make them tabu or forbidden. The idea of using forbidden regions is also adopted during the construction of the initial solution where the idea is to avoid having facilities that are too close to each other. This is explored successfully by Gamal and Salhi (2001) for the multi-source Weber problem (MSWP). In brief, during the construction phase of the initial solution or during the local search, any new continuous location, which lies in these forbidden regions, will be excluded from being selected. This useful information guides the search by avoiding these specified non-promising areas, thus reducing unnecessary computational efforts that would have been wasted otherwise.

In our experiments, a forbidden region is defined as the area enclosed by a circle with its centre at an already chosen location. The radius of the k^{th} forbidden area $(\tilde{R}_k)$ is defined as $\tilde{R}_k = \alpha R_k$ with R_k being the radius of its original circle and parameter α set close to zero, say $\alpha = 0.05$. This setting could also be made adaptive by increasing α (say $\alpha \rightarrow 2\alpha$) or decreasing it (say $\alpha \rightarrow \alpha / 2$) depending on whether the number of rejections is low or high, respectively, as demonstrated by Luis *et al.* (2009) for the capacitated MSWP. Elshaikh (2014) adapted the reformulation local search (RLS) which was originally proposed by Brimberg et al., (2014) for the MSWP by incorporating forbidden regions and other attributes. For example, when tested on the $n = 439$ *TSPLib* instance with $p = 10, 20, ..., 100$, the best average deviation is reduced from 3.114% to 2.647%, besides guaranteeing several optimal solutions, especially when $p \leq 40$. In brief, RLS is a new local search that aims to shift between discrete and continuous space while augmenting the discrete problem with the newly found continuous points (see Brimberg *et al.* (2014, 2017) for more details).

4.3 Optimal and Relaxation-based algorithms

We concentrate on two types of algorithms namely the optimal method of Drezner (1984a) and the reverse relaxation technique of Chen and Chen (2009). Neighbourhood reductions are designed for each of these algorithms making them more effective and suitable for solving larger instances either optimally or by providing tight lower bounds which could also be used for benchmarking purposes if necessary.

4.3.1 Drezner's optimal algorithm

Drezner (1984a) designed an optimal algorithm based on the idea of Z-maximal circles. A circle is defined as maximal based on a given upper bound, Z.

Let the number of potential circles with n demand points be defined by $N_c(n) = |\prod_1(n)| + |\prod_2(n)| + |\prod_3(n)|$ with

$\prod_1(n)$: set of null circles $(|\prod_1(n)| = C_1^n = n)$,

$\prod_2(n)$: set of circles made up of 2 critical points $(|\prod_2(n)| = C_2^n = n(n-1)/2)$, and

$\prod_3(n)$: set of circles made up of 3 critical points making an acute triangle

$(|\prod_3(n)| \leq C_3^n = n(n-1)(n-2)/6)$

Also let

E_j be the subset of customers encompassed by circle C_j and $r(E_j)$ its radius.

C_j is said to be Z-maximal (or maximal for short) if its radius $r(E_j) < Z$ and for any demand point $i \notin E_j; r(E_j \cup \{i\}) \geq Z$.

First, for a given value of Z the set of maximal circles is defined by

$$\Phi_Z = \{C_j;\ j = 1,..., N_c(n) \mid r(E_j) < Z \wedge r(E_j \cup \{i\}) \geq Z\ \forall i \notin E_j\}\ .$$

Drezner proposed two approaches; one uses the set covering problem while the other is based on a feasibility problem. In the former, the problem is similar to the SCP given in Subsection 2.1.2 except that for a given Z, the set of potential circles becomes Φ_Z,

The feasibility problem $(P_F(Z))$ on the other hand is defined as:

Find $X_j \in \{0,1\}$, $C_j \in \Phi_Z$, such that $\sum_{C_j \in \Phi_Z} a_{ij} X_j \geq 1\ \forall i = 1,....,n$ and $\sum_{C_j \in \Phi_Z} X_j = p$

With

$$X_j = \begin{cases} 1 \text{ if maximal circle } C_j \text{ is chosen} \\ 0 \text{ otherwise} \end{cases}$$

and $a_{ij} = \begin{cases} 1 \text{ if customer } i \text{ is encompassed by maximal circle } C_j \text{ (i.e., } i \in E_j) \\ 0 \text{ otherwise} \end{cases}$

For both formulations when the solution is found, the new value of Z is set to $Z = Max(r(E_j) \mid X_j = 1)$ which then defines the new Φ_Z. The process continues until there is no feasible solution leading to the optimal solution having the last value of Z. It was found empirically that the use of $(P_F(Z))$ is much more effective than using the set covering problem. For instance, very recently Callaghan *et al.*, (2017) showed that for the $n = 439$ *TSPLib* instance with $p = 90$, the optimal solution was found in just below 3 hours (and using 393 calls to the model) using $(P_F(Z))$, whereas the SCP-based method needed about 38 hours and 4580 calls. This observation was noted even more emphatically for the $n = 575$ *TSPLib* instance where the optimal solution was obtained after 30 hours using $P_F(Z)$, whereas the former stopped after 2 days of running with one feasible solution only and a 20% gap from the optimal solution. Callaghan *et al.* (2017) were able to speed up considerably the optimal method of Drezner (1984a), namely, the one using $P_F(Z)$. This was achieved by incorporating efficient neighbourhood reduction mechanisms thereby enabling several larger instances to be solved for the first time to optimality.

To respond to this challenge, a look-alike *p*-centre formulation $P_{op}(Z)$, which also considers $(P_F(Z))$, is first proposed.

$$\text{Minimize } R$$
$$\text{s.t. } \left\{ \sum_{C_j \in \Phi_Z} a_{ij} X_j \geq 1 \ \forall i = 1,....,n \ ; \sum_{C_j \in \Phi_Z} X_j = p; X_j \in \{0,1\}; C_j \in \Phi_Z \right\} \equiv (P_F(Z))$$
$$\text{and } X_j r(C_j) \leq R \text{ for all } j \in \Phi_Z$$

When testing the $n = 439$ *TSPLib* instance for $p = 90$ using $P_{op}(Z)$ the optimal solution was obtained about 7 times faster than with $P_F(Z)$. Though it is relatively harder to solve $P_{op}(Z)$ than $P_F(Z)$, the former produces tighter Z values leading to less calls, each requiring the long computational burden in defining Φ_Z. These results demonstrate that though a reduction in computing time is achieved, there are two issues that could help to speed up the search. These include

(i) An efficient identification of Φ_Z from one iteration (or call) to the next and

(ii) A scheme to find a good compromise solution as a feasible solution may not reflect the quality of the solution while an optimal solution may take too long to find.

The following neighbourhood reductions aim to respond to (i) and (ii).

(a) To check if C_j is Z-maximal we normally need to determine for every demand point i if $r(E_j \cup \{i\}) \geq Z$ by using the EH algorithm or similar. However, in our situation if at any iteration k of EH the radius found, say $r(E_j \cup \{i\})_k \geq Z$ then we exit EH and check for the next i. There is no need to complete EH until the end as $r(E_j \cup \{i\}) > r(E_j \cup \{i\})_k \geq Z \ \forall k$. In addition, when EH is applied, in our situation, the search starts with the critical points forming circle C_j instead. This implementation cuts the computational burden considerably. These critical points are stored in a data structure when determining circles in $\prod_1, \prod_2$ and $\prod_3$ at the very beginning so no extra computational time is really required.

(b) It is also observed that a large number of Z-maximal circles remain maximal from one iteration to the next. For instance, for $n = 439$ *TSPLib* and $p = 100$, on average less than 20% of the circles need to be tested at each iteration (Callaghan, Salhi and Nagy, 2017). It is therefore crucial to identify these circles as quickly as possible. Let Z_t be the value of Z at iteration t; then a maximal circle at iteration t remains maximal at iteration $t+1$ if $r(E_j) < Z_{t+1}$ according to Lemma 1 in Callaghan *et al*. (2017). This leads to not checking the expensive part which is $r(E_j \cup \{i\}) \geq Z_{t+1} \ \forall i$ as $Z_{t+1} < Z_t$.

(c) It is important to detect whether a circle is maximal or not quickly so as to avoid performing an unnecessary full check. According to Lemma 2 in Callaghan *et a*l. (2017), if $\exists i \notin E_j \mid d(P_i, CT_j) < Z$ then C_j is not Z-maximal. Also there are some points that do not need to be checked. For instance if $i \notin E_j$ and $d(P_i, CT_j) \geq 2Z$ there is no need to find $r(E_j \cup \{i\})$ as $r(E_j \cup \{i\}) > Z$ (see Lemma 3 in Callaghan *et al.*, 2017). In other words, if $d(P_i, CT_j) \geq 2Z \ \forall i \notin E_j$, C_j is systematically Z-maximal. This leads to performing the check for those points in $\{i \notin E_j \mid Z \leq d(P_i, CT_j) < 2Z\}$ only. If this set is empty and

Lemma 2 does not apply, then C_j is Z -maximal. In other words, there is no need to check it.

(d) It is also useful to identify information from previous non-maximal circles. For instance, if a circle C_j at iteration t is found non-maximal, it means there was at least one point $i_1 \notin E_j$, say the qth point to be evaluated, that led to $r(E_j \cup \{i_1\}) < Z_t$. In the next iteration, it is important to start with i_1 to check whether or not $r(E_j \cup \{i_1\}) < Z_{t+1}$. This means the first $(q-1)$ points are ignored leading to a saving of $(q-1)$ unnecessary checks each involving the use of EH or its equivalent. This is considerable when applied to all circles.

For example, when these neighbourhood reduction schemes (a-d) are implemented for the $n = 439$ *TSPLib* instance for $p = 70, 80, 90$ and 100, a massive reduction in computational time is recorded. Individually (c) yields about 84% reduction, followed by (d) with a similar amount of 83%, with (a) producing just below 51% and finally (b) resulting in 26%. When all four are combined together following the ranked order of their individual performances *c-d-a-b*, the following cumulative percentage reductions of 84%, 90%, 96% and 97% are recorded. This shows that only a tiny 3% of the total time is required there by demonstrating the power of these neighbourhood reductions, which also enable the enhanced algorithm to solve to optimality larger instances.

To tackle (b) a detailed analysis showed that CPLEX consumes approximately 30% and 80% on average of the CPU time (Callaghan *et al*., 2017) for $n = 439$ *TSPLib* and $n = 575$ *TSPLib* , respectively. The higher values are found with larger values of p, reaching 99% for the largest problem. This was also noted to occur at the latter iterations mainly to guarantee optimality of $P_{op}(Z)$ at a given Z. In order to alleviate this issue, a scheme that adaptively guides the level of usage of CPLEX is added. This scheme aims to terminate CPLEX earlier if a compromise solution is considered to be good. To achieve this, a moving average over the last m iterations is recorded for both the computing time for identifying the maximal circles which we denote by $CPU(\max)$ and the time for running CPLEX denoted by $CPU(cplex)$. We define the ratio of these two times as $\lambda = \dfrac{CPU(\max)}{CPU(cplex)}$. If $\lambda \geq 1$, this shows that the time for identifying maximal circles is relatively higher. In this case, we solve the

problem to optimality. However, if it is not the case, we set two additional levels for the duality gap as $GAP(\%) = \begin{cases} 1.0 & \text{if } \lambda \le 0.4 \\ 0.5 & \text{if } 0.4 < \lambda < 1 \\ 0 & \text{otherwsie} \end{cases}$

The above reduction schemes have contributed significantly in determining several optimal solutions for large instances up to $n = 1323$ and $p = 10, 20, ...100$ for the first time.

4.3.2 Relaxation-based approaches

The idea is to relax the original problem by successively solving small sub problems that gradually increase in size until an optimal solution is found. Handler and Mirchandani (1979) originally discussed this idea of relaxation, but Chen and Handler (1987) proposed an algorithm where at each iteration, a demand point is added and the new augmented sub problem is then solved again. The search continues until an optimal solution for the sub problem happens to be feasible for the original problem. Chen and Chen (2009) revisited the problem by adding k demand points at a time. Three relaxation-based algorithms known as the improved, the binary and the reverse relaxation algorithms were presented. Callaghan (2016) performed an extensive experiment and concluded that the reverse relaxation algorithm is the most promising. This led to the design of three neighbourhood reductions to speed up this algorithm so it can be used to solve larger instances either optimally or by providing tight lower bounds (Callaghan et al., 2018). For convenience, the reverse relaxation algorithm is briefly summarised in Figure 5 as some of its steps form the basis of the following neighbourhood reductions.

a- In Step 1, the initial subset is randomly chosen which may not be easy to replicate and may lead to either fast or slow convergence. One way forward is to construct such a subset deterministically reflecting the characteristics of the p-centre problem. As shown by Chen and Handler (1987), the smallest possible value of $|SUB|$ required to yield a solution of p circles is to have at least

$$Min_{SUB} = \underset{r \ge 3}{Min}\, N_c(r) \ge p \text{ with } N_c(r) = C_1^r + C_2^r + C_3^r \quad .$$

1- Set the lower bound LB, the value of k, the set of potential circles J ($|J| = N_c$) and choose randomly a subset of demand points SUB ($|SUB| << n$).

2- Compute a_{ij} ($i = 1,...,|SUB|$; $j = 1,...,N_c$) based on LB and solve the corresponding set covering problem.

3- If the solution X is feasible (i.e., $\sum_{j \in J} X_j \leq p$) go to step 4;

Otherwise set the new value of LB to the smallest radius of a circle in SUB that is larger than LB, and go to Step 2.

4- If X is feasible for the original problem, the optimal solution is X and stop.

Otherwise add k furthest demand points to SUB and go to step 2.

Figure 5: Main steps of Chen and Chen's algorithm (2009)

The idea is to use the vertices of the convex hull as a guide and let CH denote such a subset. It can be shown that these points are not all necessarily critical points. We construct SUB as follows: Let i_1 be the furthest point to CH and set $SUB = \{i_1\}$. If $|SUB| < p$, allocate all demand points to their nearest point in SUB and identify the largest cluster. Choose the next point to add to SUB as the furthest point from this largest cluster, say i_2 and set $SUB = SUB \cup \{i_2\}$. This mechanism is repeated until $|SUB| = Min_{SUB}$ where the construction of the circles is performed. If the solution is not feasible in SUB (i.e., there are not enough circles) , continue the addition of new points in the same way until a feasible solution is found.

b- In Step 4, the added k points need not be necessarily the furthest points. For instance in the worst scenario, all or most of the furthest k points may belong to the same elongated cluster as shown in Figure 6 where four points (P_1, P_2, P_3, P_4) are close to each other forming a small cluster, denoted by cluster 1. Once P_1 (the furthest from the solution) is added (in bold), a new much improved solution can be found showing that its contribution is important. However, the addition of the other three points, P_2, P_3 and P_4, will not affect this new solution and their inclusion will only add unnecessary computations. It is therefore important to identify such a cluster so these

three points do not need to be part of *SUB*. This rule can be applied to other clusters to identify the more representative points to add. The addition of the k new points can be performed either by including one point at a time followed by the evaluation of the new solution configuration or by adding all the k points in one go.

To speed up the search even more, artificial circles are constructed whose centres remain the centres of the existing circles but whose radii are increased to Z. These are the dotted circles shown in Figure 6. The checking is based on those uncovered points away from the artificial circles instead of using all the initial uncovered points. As an example when tested on the $n = 439$ *TSPLib* instance for $p = 10, 20, ..., 100$, this neighbourhood reduction eliminates over 10% of computing time.

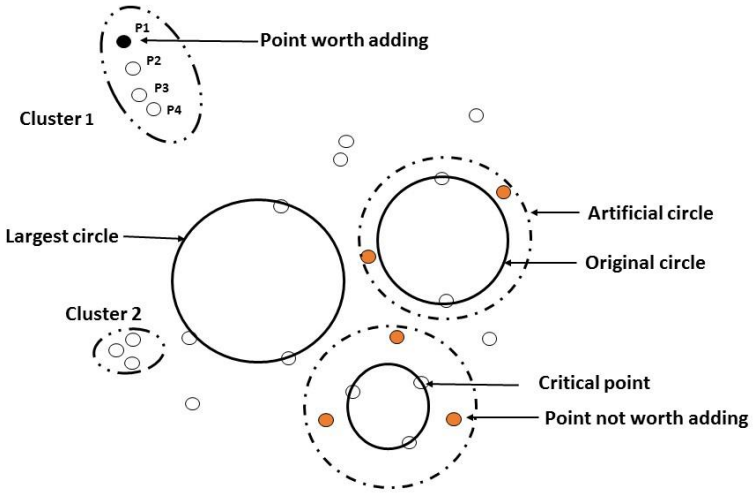

Figure 6: Effect of point clusters and artificial circle in the addition of the new points

c- Also, the value of k in Step 4 does not have to remain constant at each iteration and for all instances. This parameter can be made dynamic at a given iteration t. Let N_{unc}^{t} denote the number of uncovered points at iteration t. We can then define $k(t) = f(p, N_{unc}^{t})$.

d- In Step 3, LB is updated by taking the next radius larger than the current value of LB. Though this is mathematically correct and will end up with the final value, the search may use too many updates many of which turn out to be unnecessary. Instead of

choosing the next largest, we adopt a jump-based scheme to select the $jump^{th}$ largest. A similar but simpler idea was initially proposed and successfully implemented for the vertex p-centre problem by Al-Khedhari and Salhi (2005) where a jump of two was used. This is extended by defining the jump as a function of a predefined maximum jump size, and the ratio $\frac{N_{unc}^{t}}{n}$. This jump-based scheme systematically learns as the search progresses. Note that the obtained solution may provide an upper bound instead of a lower bound. In this case, a backtracking is required by evaluating the values of LB between $LB(t)$ which was definitely a lower bound and $LB(t+1)$ which happens to be an upper bound. Here at most $(jump(t)-1)$ jumps may be required to guarantee optimality.

These neighbourhood reductions were found very promising when compared to the original implementation. For instance, when tested on the $n=439$ *TSPLib* instance with $p=10$, 20, …, 100, an average reduction in computing time of nearly 90%, with the smallest being just over 50% and the largest nearly 97%, was observed.

The incorporation of all the above reduction schemes enables the algorithm to solve most of the larger instances optimally within 3 hours of computing time. For those instances where the optimal solution could not be guaranteed, a tight lower bound was recorded, which may be used in the future for assessing new heuristics.

5 Neighbourhood reduction highlights and conclusions

In this chapter, a brief review of both the vertex and planar *p*-centre problems is given with an emphasis on contributions made by Zvi Drezner. Several neighbourhood reductions especially designed for these two location problems are then discussed with the aim to enhance the efficiency of existing algorithms or in assisting at designing a more effective heuristic or optimal algorithm. For the vertex $p-$centre problem, a series of neighbourhood reduction rules are presented that have enhanced the performance of existing optimal algorithms considerably thus enabling the exact solution of well-known ORLib instances $(n=100$ to $900)$ and the $n=1060$ TSPLib instance to be obtained in much faster time than

before. Similarly, for the continuous p -centre problem, four TSPLib instances varying in size from $n = 439$ to 1323 with $p = 10, 20, ..., 100$ are used as a platform to demonstrate the effectiveness of the proposed neighbourhood reduction schemes. Enhanced VNS and perturbation heuristics are now much more effective than before. Also, Drezner's optimal algorithm and the relaxation-based methods of Chen and Chen are now able to provide optimal solutions for the first time for many of the largest instances tested, and tight lower bounds for the rest.

The implementation of these exact methods using the neighbourhood reduction schemes discussed in this chapter can be made even faster if a tighter initial solution is provided, say by a powerful metaheuristic. In addition, as these schemes tend to cut on computational time by avoiding time wastage, if the same allowed computing time is used as the stopping criterion for the enhanced version, the new solution might easily improve on the original one as many more iterations would be performed leading to more moves being evaluated.

It is also interesting to observe that in Subsection 4.3.1(c), the checking area in Lemma 3 and the recording of the points that define non maximal circles can be made slightly tighter as recently pointed out by Plastria (2017).

There exist a few variations of the *p*-centre problem. In the conditional *p*-centre problem some (say *q*) facilities already exist and the objective is to locate *p* new facilities in addition to the existing *q* facilities. Minieka (1980) presented the problem while Drezner (1989, 1995) defined it formally as the (*p, q*) centre problem, and put forward a binary search to solve it. Chen and Chen (2010) also adapted their algorithm discussed in Subsection 4.3 to tackle this problem. Another related problem is when each demand point needs to be covered by at least α facilities. This problem, initially proposed by Krumke (1995), is known as the α -neighbourhood *p*-centre problem, and has its applications in the case of facility disruption. Chen & Chen (2013) used Minieka's algorithm and modified their relaxation method described in Subsection 4.3 to solve this problem. Very recently, Callaghan, Salhi and Brimberg (2018) studied both variants by adapting the powerful reduction schemes discussed in Subsection 4.3 so that larger problems can now be solved to optimality for the first time.

One possible extension is to adapt the neighbourhood reductions used for the continuous problem in Subsection 4.3 that rely on maximal circles, cluster points and artificial circles, to

the discrete problem though this case can be solved by other means. It is also worth noting that reduction schemes do exist for other combinatorial and global optimisation problems. In general the more constrained the problem is, the more significant the impact of neighbourhood reduction can be. For example, in the vehicle routing problem, a saving on CPU time of up to 85% was recorded without a significant loss in solution quality (Salhi and Sari, 1997; Sze *et al.*,2016; Sze *et al.*, 2017).

It is necessary to mention in conclusion that the use of neighbourhood reduction techniques may adversely affect the solution quality. The aim is therefore to construct such schemes which only exclude moves that have a high probability of not harming the quality of the solution. This risk presents an exciting challenge of finding the right balance between a strong neighbourhood reduction (remove as much as possible) and maintaining solution quality.

Acknowledgment- The first author is grateful to his former PhD students, namely, Professors AI. Al-Khedhairi, AM. Alharbi, AM. Elshaikh and Drs CA. Irawan, and B. Callaghan. Their contributions in this area during the last 15 years are very much appreciated.

References

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

[2] Al-Harbi AM. (2010). Combining heuristics and exact approaches for the vertex p-centre problem and other related location problems, *PhD Dissertation*, University of Kent, Canterbury, UK.

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

[4] Brimberg J, Drezner Z, Mladenović N and Salhi S. (2017). Using injection points in reformulation local search for solving continuous location problems, *Yugoslav Journal of Operations Research* **27**: 291-300.

[5] Callaghan, B. (2016). An investigation into exact methods for the continuous p-centre problem and its related problems, (PhD Thesis), University of Kent, Canterbury, UK

[6] Callaghan B, Salhi S and Nagy G. (2017). Speeding up the optimal method of Drezner for the p-centre problem in the plane, *European Journal of Operational Research* **257**: 722-734.

[7] Callaghan B, Salhi S and Brimberg J. (2018). Optimal solutions for the continuous p-centre problem and related α -neighbourhood and conditional problems: A relaxation-

based algorithm, *Journal of the Operational Research Society* doi: 10.10880/01605682.2017.1421854.

[8] Chen R. (1983). Solution of minisum and minimax location-allocation problems with euclidean distances, *Naval Research Logistics Quarterly* **30**: 449-459.

[9] Chen R and Handler GY. (1987). Relaxation methods for the solution of the minimax location-allocation problem in euclidean space. *Naval Research Logistics Quarterly* **34**: 775-788.

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

[11] Chen, D., Chen, R. (2010). A relaxation-based algorithm for solving the conditional p-center problem. Oper. Res. Lett. **38**: 215-217.

[12] Cooper L. (1964). Heuristic methods for location-allocation problem, *SIAM Review* **6**: 37-53.

[13] Chen, D., Chen, R. (2013). Optimal algorithms for the α -neighbor *p*-center problem. *European Journal of Operational Research* **225**: 36-43.

[14] Daskin, M.S. (2000). A new approach to solving the vertex *p*-center problem to optimality: algorithm and computational results. Commun. Oper. Res. Soc. Jpn. **45**: 428–436.

[15] Drezner Z. (1984a). The p-center problem- heuristic and optimal algorithms. *Journal of the Operational Research Society* **35**: 741-748.

[16] Drezner Z. (1984b). The planar two-center and two-median problems, *Transportation Science* **18**: 351-361.

[17] Drezner, Z. (1989). Conditional *p*-center problems. *Transp. Sci*. **23**: 51-53.

[18] Drezner, Z. (1995). On the conditional p-median problem. *Comput. Oper. Res*. **22**: 525-530.

[19] Drezner Z. (2011). Continuous Center Problems. in *Foundations of Location Analysis,* In (H.A.Eiselt and V. Marianov, Eds); Springer-Verlag, pp 63-78.

[20] Elloumi, S., Labbe, M., Pochet, Y. (2004). A new formulation and resolution method for the *p*-center problem. INFORMS J. Comput. **16**: 84–94.

[21] Elshaikh A. (2014). Adaptive heuristic methods for the continuous p-centre problem, *PhD Dissertation;* University of Kent, Canterbury, UK.

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

[23] Elshaikh A., Salhi S., Brimberg J., Mladenovic N., Callaghan B., and Nagy G. (2016). An adaptive perturbation-based heuristic: An application to the continuous p-center location problems, *Computers & Operations Research* **75**: 1-11.

[24] Eiselt HA and Charlesworth G. (1986). A note on *p*-center problems in the plane, *Transportation Science* **20**: 130-133.

[25] Eiselt HA and Sandblom CL. (2004). Decision analysis, location models, and scheduling problems, Springer, NY.

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

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

[28] Handler, G.Y., and Mirchandani, P.B. (1972). Locations on networks: theory and algorithms. Cambridge, MA: MIT Press

[29] Hansen P, Mladenovic N, Brimberg J and Moreno-Perez JA. (2010). Variable

neighbourhood search, in *Handbook of Metaheuristics*, (Gendreau M and Potvin J-Y , Eds), chapter 3, 61-86, (2nd edition), Springer, NY.

[30] Hakimi, S.L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. Oper. Res. **12**: 450-459.

[31] Hearn, D.W. and Vijay, J. (1982). Efficient algorithms for the (weighted) minimum circle problem, Operations Research 30, 777-795.

[32] Ilhan, T., Pinar, M. (2001). An efficient exact algorithm for the vertex *p*-center problem. URL (http://www.ie.bilkent.edu.tr/~mustafap/pubs/).

[33] Irawan CA., Salhi , S. and Drezner Z. (2016). Hybrid metaheuristics with VNS and exact methods: application to large unconditional and conditional vertex p-centre problems. *Journal of Heuristics* **22**: 507-537.

[34] Kariv, O. and Hakimi, S.L. (1979). An algorithmic approach to network location problem, I: The p-centers, SIAM Journal on Applied Mathematics 37, 539-560.

[35] Krumke, SO. (1995). On a generalization of the p-centre problem, *Information Processing Letters* **56**: 67-71.

[36] Luis, M., Salhi, S. and Nagy G. (2009). Region-rejection based heuristics for the capacitated multi-source Weber problem. *Computers & Operations Research* **36**: 2007-2017.

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

[38] Mladenović N and Hansen P. (1997). Variable neighbourhood search. *Computers & Operations Research* **24**: 1097-1100.

[39] Mladenović N, Labbe M and Hansen P. (2003). Solving the p-center problem with tabu search and variable neighbourhood search, *Networks* **42**: 48-64.

[40] Minieka, E. (1970). The m-center problem, *SIAM Review* **12**:138-139.

[41] Minieka, E. (1980). Conditional Centers and Medians on a Graph. *Networks* **10**, 265-272.

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

[43] Plastria F. (2017). Private communication.

[44] Salhi S. (2017). *Heuristic Search: The Emerging Science of Problem Solving*, Palgrave McMillan (published by Springer); Chan, Switzerland.

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

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

[47] Salhi S and Sari M. (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem, *European Journal of Operational Research* **103**: 95-112.

[48] Sze JF, Salhi S and Wassan N. (2016). A hybridisation of adaptive variable neighbourhood search and large neighbourhood: application to the vehicle routing problem. *Expert System with Applications* **65**: 383-397.

[49] Sze JF, Salhi S and Wassan N. (2017). The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive search and large neighbourhood search. *Transportation Research Part B* **101**: 162-184.

[50] Xu S, Freund R and Sun J. (2003). Solution methodologies for the smallest enclosing circle problem. *Comput Optim Appl* **25**: 283-292.