PARTICLE SWARM OPTIMIZATION FOR P-MEDIAN PROBLEMS

by

Ruslan MAMEDSAIDOV

July 2009

PARTICLE SWARM OPTIMIZATION FOR P-MEDIAN PROBLEMS

by

Ruslan Mamedsaidov

A thesis submitted to

the Graduate Institute of Science and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

July 2009
Istanbul, Turkey

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____
Assist. Prof. Mehmet SEVKLI
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____
Assist. Prof. Mehmet SEVKLI
Supervisor

Examining Committee Members

Assist. Prof. Mehmet SEVKLI _____

Assist. Prof. Fatih CAMCI _____

Assist. Prof. Fahrettin ELDEMIR _____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

_____
Assoc. Prof. Dr. Nurullah ARSLAN
Director

Date
July 2009

PARTICLE SWARM OPTIMIZATION FOR P-MEDIAN PROBLEMS

Ruslan Mamedsaidov

M. S. Thesis – Industrial Engineering
July 2009

Supervisor: Assist. Prof. Mehmet Şevkli

ABSTRACT

In this work, a discrete particle swarm optimization algorithm (DPSO) is proposed for the p-median problem. Although the algorithm has all major characteristics of the classical particle swarm optimization (PSO), the search strategy of the algorithm is different. The algorithm is applied to the p-median problem with the objective of minimizing distance between demand points and facilities. A novel proposed continuous particle swarm optimization (NCPSO) algorithm for p-median problem is introduced as well. The results of both algorithms are compared against each other. And the performance of proposed DPSO is compared with against other algorithms in literature, Neural model, Reduced Variable Neighborhood Search and Simulated Annealing. The experiments have shown that the proposed algorithm results in better performance.

Keywords: Swarm Intelligence, Particle Swarm Optimization, P-Median Problem, NP-hard.

P-MEDİAN PROBLEM İÇİN PARÇACIK SÜRÜ OPTİMİZASYONU

Ruslan Mamedsaidov

Yüksek Lisans Tezi – Endüstri Mühendisliği
Temmuz 2009

Tez Yöneticisi: Yrd. Doç. Dr. Mehmet Şevkli

ÖZ

Bu tezde, p-median problem için yeni bir parçacık sürü optimizasyonu (DPSO) önerilmiştir. Önerilen algoritmada klasik Parçacık Sürüsü Optimizationunun (PSO) bütün karakteristikleri olmasına rağmen, algoritmanın arama stratejisi farklıdır. Algoritma, talep noktaları ve tesislerin arasında mesafeyi en aza indirme amacı ile p-median problemine uygulanmıştır. Bunun dışında literatürde bulunan Sürekli Parçacık Sürü optimizasyonundan farklı bir Sürekli Parçacık Sürü optimizasyonunu önerilmiştir. Önerilen iki algoritmanın sonuçları literatürde bulunan başka algoritmaların sonuçları ile karşılaştırılmış ve daha iyi olduğu görülmüştür.

Anahtar Kelimeler: Sürü Zekası, Parçacık Sürü Optimizasyonu, p-Median Problemi, NP-Zor

To my Family,

ACKNOWLEDGMENT

I express sincere appreciation to Assist Prof. Mehmet ŞEVKLİ for his guidance and insight throughout the research.

Thanks go to Assist. Prof. Fahrettin ELDEMİR, Assist. Prof. Fatih Camcı and PhD. Student Recep Kızılaslan for their valuable suggestions and comments.

I express my thanks and appreciation to my family for their understanding, motivation and patience. Lastly, but in no sense the least, I am thankful to all colleagues and friends who made my stay at the university a memorable and valuable experience.

TABLE OF CONTESTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCION

1.1. Discrete Location Problems

Discrete Optimization has set up as an important component in modern applied mathematics. Many problems from business and industry can be modeled as discrete optimization problems. Discrete location problems typically involve a finite set of sites at which supply points can be located, and a finite set of clients to be fulfilled by the supply points which demands for service or good. The most well known discrete location problems are the Uncapacitated (and Capacitated) Facility Location Problem, Multi Stage Uncapacitated Facility Location Problem. Evidently many extensions of these basic location problems have been developed depending on the objective function. We want to state that the main differentiator in the most classical location models is the objective function. A great variety of objective functions have been considered, for instance a *median* objective where the task is to minimize the sum of the costs of fulfilling all the demand requests from the clients. The *center* objective is to minimize the maximum cost of fulfilling all the demand requests from the clients, using the sites chosen. The convex combination of the above mentioned median and center objectives is considered as well and can be called as a *centdian* objective where the aim is to keep both the average cost behavior as well as the highest cost in balance. So the well known *p*-median Problem and *p*-center Problem are defined. Below we are going to show briefly some of the Discrete Location Problems.

1.1.1. Uncapacitated Facility Location Problem

Simple Plant Location Problem is another title for Uncapacitated Facility Location Problem. It is assumed that in Uncapacitated Facility Location Problem given a set $I = \{1 \ldots N\}$ potential facility locations by providing some uniform product. A facility can be opened in any location $i \in I$, opening a facility at location i has nonnegative cost as well as the transportation cost of satisfying the customer requirements from a facility. Each open facility can provide an unlimited amount of products.

Let a set $J = \{1 \ldots M\}$ assign customers that require service. For each pair (i, j) is given the process or transportation costs $g_{ij} \geq 0$. The goal is to determine a subset of the set of potential facility locations $S \subseteq I$, $S \neq \varnothing$, at which to open facilities and an assignment of all clients to these facilities so as to minimize the overall total cost. The problem can be written as the following:

$$F(S) = \sum_{i \in S} C_i + \sum_{j \in J} \min_{i \in S} g_{ij} \to \min_{S \subseteq I} \qquad 1.1$$

Stated problem is the generalization of the well-known set covering problem and, therefore, it is NP-hard problem (G. Cornuéjols et al, 1990). Exact algorithms, approximation algorithms with constant performance guarantee, Lagrangian heuristics, Particle Swarm Optimization (Guner and Sevkli, 2008) and randomized iteration algorithms of local search were developed for solving simplest location problem. Polynomially solvable classes of problem were found. The reader can find the review of results for this problem, for example, in (Mirchandani et al. 1990).

1.1.2. Capacitated Facility Location Problem

Capacitated Facility Location Problem (CFLP) is generalization of the Uncapacitated Facility Location Problem. The main differentiator is the very important assumption that In contrast to Uncapacitated Facility Location Problem we now suppose each facility can provide a limited number of production. And although mathematical models of those

problems do not vary too much, solving methods for CFLP are more difficult. There has been done much related to this problem. (Sridharan R, 1995)

Now we describe the mathematical model as integer programming problem. Let a set $I=\{1,..., \boldsymbol{I}\}$ give potential facility locations by providing some uniform product. The number $c_i \geq 0$ is the opening cost of facility at location $i \in I$, $V_i \geq 0$ is the maximum value of production the facility can provide.

A set $J=\{1,\ldots,\boldsymbol{J}\}$ assign customers that require service. For each pair i,j $g_{ij} \geq 0$ is the production and transportation costs and $p_{ij} \geq 0$ is a value of product from facility i needed to client j.

Let us define the following notations:

$$y_i = \begin{cases} 1 & \text{if facility } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \qquad 1.2$$

$$x_{ij} = \begin{cases} 1 & \text{if client } j \text{ is serviced by facility } i, \\ 0 & \text{otherwise} \end{cases} \qquad 1.3$$

Then the Capacitated Facility Location Problem may be written as:

$$\sum_{j \in J} p_{ij} x \leq V_i y_i, \quad i \in I, \qquad 1.4$$

$$\min \left\{ \sum_{i \in I} c_i y_i + \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \right\} \qquad 1.5$$

$$\sum_{i \in J} x_{ij} = 1, \quad j \in J, \qquad 1.6$$

$$x_{ij}, \quad y_i \in \{0,\ 1\},\ i \in I,\ j \in J \qquad 1.7$$

1.1.3. Multi Stage Uncapacitated Facility Location Problem

In the Multi Stage Uncapacitated Facility Location Problem we are given a set of facilities and a set of customers. Each customer must be serviced by a sequence of different facilities. These sequences are defined by hierarchy of production and distribution system and can be presented as facility paths. The set of admissible facility paths is given. Each facility has fixed cost. Each customer has transportation costs for servicing by the facility paths. The problem is to select facilities in order to service the customers with minimal total cost.

Below we present an illustrated example of hierarchical multi stage production system. A feasible solution of the problem is marked in black.

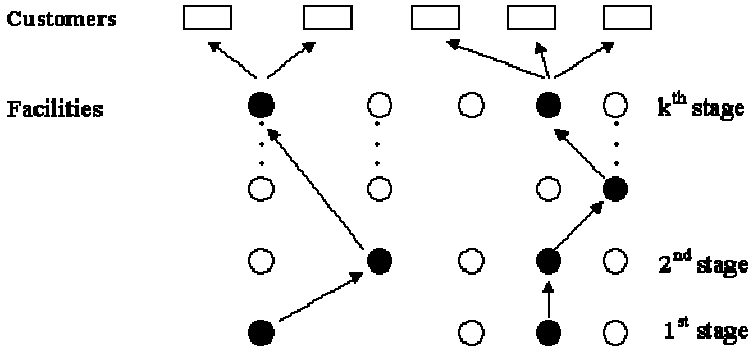

Figure 1.1 Hierarchical multi stage production system. (Sobolev Institute of Mathematics, 2009)

1.1.4. P-Center Problem

The p-center problem, in other words the *minimax* problem, aims to find the location of *p* number of facilities (centers) on a network so that the maximum distance traveled from each customer (demand point) to its nearest facility minimized. (Daskin, 1995; Kariv and Hakimi, 1979a). This problem may address, for instance, the location of public facilities, schools, emergency services, where the generally accepted objective is to design a system

so that no customer has to travel too far (or each customer could be reached in a reasonable amount of time).

In this work we concentrated much more on the problem which can be described as a *minisum*, the p-median problem.

1.2. The Particle Swarm Optimization

Particle swarm optimization (PSO) is based on the metaphor of social interaction and communication among different spaces in nature, such as bird flocking and fish schooling. It is different from other evolutionary methods in a way that it does not use the genetic operators (such as crossover and mutation), and the members of the entire population are maintained through out the search procedure. Thus, information is socially shared among individuals to direct the search towards the best position in the search space. In a PSO algorithm, each member is called a particle, and each particle moves around in the multi-dimensional search space with a velocity constantly updated by the particle's experience, the experience of the particle's neighbors, and the experience of the whole swarm. PSO was first introduced to optimize various continuous nonlinear functions by (Eberhart and Kennedy, 1995). PSO has been successfully applied to a wide range of applications such as automated drilling (Onwubolu and Clerc, 2004), neural network training (Van den Bergh and Engelbecht, 2000), scheduling problems (Tasgetiren et. al., 2006), (Tseng and Liao, 2008), (Allahverdi and Al-Anzi, 2006), (Tasgetiren et. al., 2007), and (Pan et al., 2008), power and voltage control (Yoshida, 2000), and task assignment (Salman, 2003). The PSO algorithm was successfully applied to the similar problem as p-median problem which is Uncapacitated Facility Location Problem by (Guner and Sevkli, 2008). More information about PSO can be found in (Kennedy et al. 2001).

In PSO, each single solution, called a particle, is considered as an individual, the group becomes a swarm (population) and the search space is the area to explore. Each particle has a fitness value calculated by a fitness function, and a velocity to fly towards the optimum. All particles fly across the problem space following the particle that is nearest to

the optimum. PSO starts with an initial population of solutions, which is updated iteration-by-iteration. The principles that govern PSO algorithm can be stated as follows:

- n dimensional position ($X_i = (x_{i1}, x_{i2}, ..., x_{in})$) and velocity vector ($V_i = (v_{i1}, v_{i2}, ..., v_{in})$ for i^{th} particle starts with a random position and velocity.
- Each particle knows its position and value of the objective function for that position. The best position of i^{th} particle is donated as $P_i = (p_{i1}, p_{i2}, ..., p_{in})$, and the best position of the whole swarm as, $G = (g_1, g_2, ..., g_n)$ respectively. The PSO algorithm is governed by the following main equations:

$$v_{in}^{t+1} = wv_{in}^{t} + c_1 r_1 (p_{in}^{t} - x_{in}^{t}) + c_2 r_2 (g_i^{t} - x_{in}^{t}), \quad 1.8$$

$$x_{in}^{t+1} = v_{in}^{t+1} + x_{in}^{t} \quad 1.9$$

where t represents the iteration number, w is the inertia weight which is a coefficient to control the impact of the previous velocities on the current velocity. c1 and c2 are called learning factors. r1 and r2 are uniformly distributed random variables in [0, 1].

The original PSO algorithm can optimize problems in which the elements of the solution space are continuous real numbers. The major obstacle for successfully applying PSO to combinatorial problems in the literature is due to its continuous nature. To remedy this drawback, (Tasgetiren et al. 2006, 2007) presented the smallest position value (SPV) rule. Another approach to tackle combinatorial problems with PSO is done by (Pan et al., 2008). They generate a similar PSO equation to update the particle's velocity and position vectors using one and two cut genetic crossover operators.

1.2.1. Discrete particle swarm optimization

As we mentioned before Particle Swarm Optimization initially was introduced for Continuous problems. In (Kennedy and Eberhart, 1997) the modification to the algorithm was proposed by utilizing a probability value in a range of [0.0, 0.1] according to which the

position vector takes discrete values, 0 or 1. The probability value was derived through the sigmoid function:

$$s(V_i) = 1/(1 + \exp(-V_i)) \qquad 1.10$$

The position of the vector value is set to 1 if the random number (from a uniform distribution between 0.0 and 1.0) less than the value of the sigmoid function or set to 0 otherwise or vice versa, as in Equation (1.11)

$$X_i = \begin{cases} 1 & r < s(V_i) \\ 0 & otherwise \end{cases} \qquad 1.11$$

Another Discrete Particle Swarm Optimization was proposed by (Afshinmanesh et al., 2005) where the idea is to implement artificial immune system within the PSO algorithm, i.e. the theory of negative selection was used for calculation of velocity vector and the movements of the particle in the proposed technique.

The process of generating a new position vector for a selected individual in a swarm is as follows: two different particles are generated out of the particle position vector and two "desired positions" (the global best position vector and individual's best position vector). It is done using an XOR logical operation, as in Equations (11) and (12). Each bit in the developed vector shows whether this bit is different from the desired one or not (i.e. the Hamming distance). The "exploration ability", diversity, has been added into the method by creating two random discrete vectors consisting 0 or 1 values that will be processed with previously generated vectors by an 'and' logical operation. The velocity vector, which shows which bits should be changed, is produced by performing an 'or' operation between the two vectors produced as a result of the 'and' operation in the previous step, as in Equation (13). As a result, a new position vector will be computed by applying an 'xor' logical operation to the velocity vector and the particle's current position vector, shown in Equation (14). The negative selection was applied just at this stage for the reason that the number of ones and zeros are predefined. So if the number of ones exceeds predefined value the negative selection is applied to convert ones into zeros.

Application of Discrete Particle Swarm Optimization to condition-based maintenance and comparison with genetic algorithm was analyzed in (Camci, F., 2009)

Below is given the algorithm expressed in formulas and the illustration for the proposed algorithm.

$$d1_{i,t} = XOR\left(Pbest_i, X_{i,t}\right) \tag{1.12}$$

$$d2_{i,t} = XOR\left(Gbest_i, X_{i,t}\right) \tag{1.13}$$

$$V_{i,t+1} = OR\left(AND\left(rand\left(1,m\right), d1_{i,t}\right), AND\left(rand\left(1,m\right), d2_{i,t}\right)\right) \tag{1.14}$$

$$X_{i,t+1} = XOR\left(X_{i,t}, V_{i,t+1}\right) \tag{1.15}$$

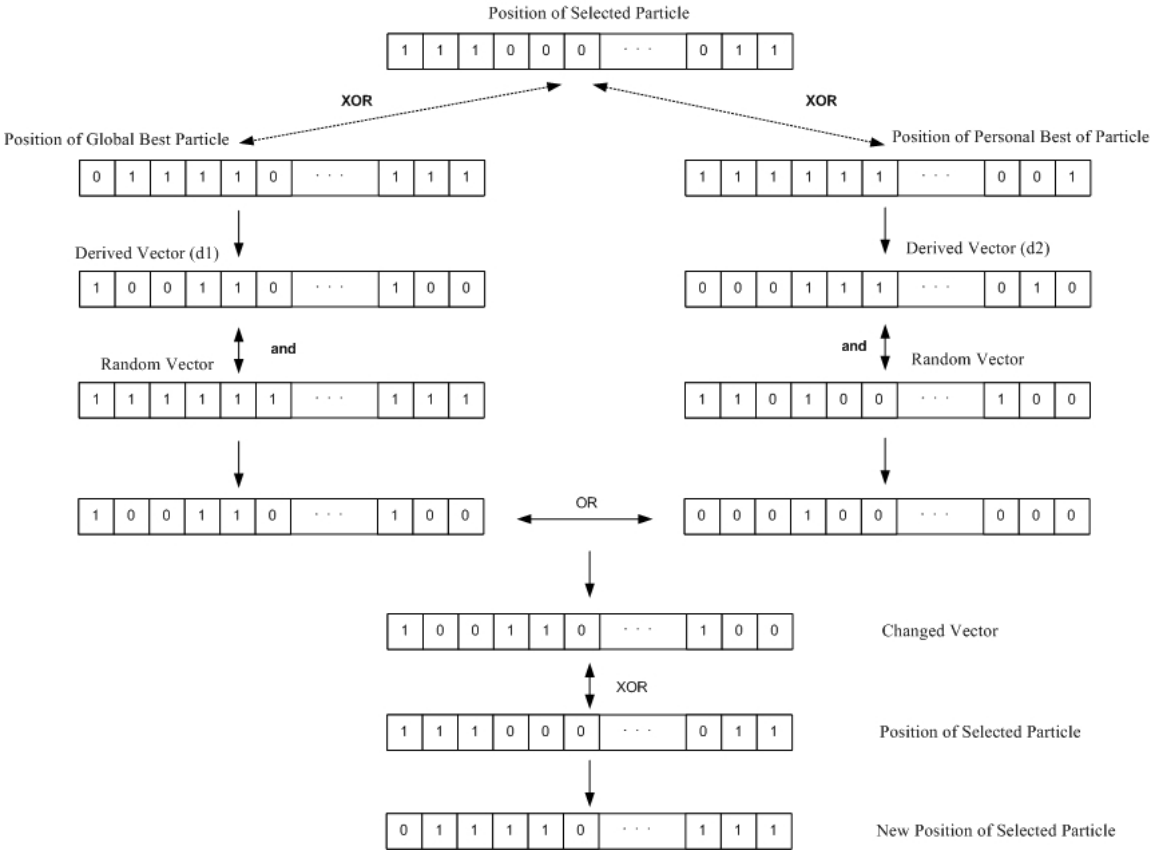

Figure 1.2 Individual generation in proposed algorithm (Afshinmanesh et al., 2005)

CHAPTER 2

THE P-MEDIAN PROBLEM

2.1. Basic Problem Definition

P-median problem is a well known facility-location problem where the task is to allocate p facilities in a way that the total distance between *n* demand points and *p* facilities minimized. It is shown by Kariv and Hakimi that the p-median problem is NP-hard (Kariv and Hakimi, 1979b).

Mathematical formulation of p-median problem is as follows, (ReVelle and Swain, 1970).

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} a_i d_{ij} x_{ij} \qquad 2.1$$

Subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i=1, 2, \ldots, n, \qquad 2.2$$

$$x_{ij} \le y_j, \qquad i, j=1, 2, \ldots, n, \qquad 2.3$$

$$\sum_{j=1}^{n} y_j = p \qquad 2.4$$

$$x_{ij}, y_j \in \{0,1\}, \qquad i, j=1, 2, \ldots, n, \qquad 2.5$$

where

n = total number of nodes in the graph,

a_i = demand of node i,

d_{ij} = distance from node i to node j,

p = number of facilities used as medians

a_i, d_{ij} are positive real numbers,

$$x_{ij} = \begin{cases} 1 & \text{if node } i \text{ is assigned to facility at point } j, \\ 0 & \text{Otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{If facility is locate at point } j, \\ 0 & \text{Otherwise} \end{cases}$$

2.2. Literature review

P-median problem is a well known facility-location problem where the task is to allocate p facilities in a way that the distance between *n* demand points and facilities minimized. It is shown by Kariv and Hakimi that the p-median problem is NP-hard (Kariv and Hakimi, 1979b). It is unlikely to obtain optimal schedule through polynomial time-bounded algorithms. Small size instances of p-median problem can be solved with reasonable computational time by exact algorithms such as branch-and-bound (Järvinen et al., 1972). Where the method works by finding (n – p) nodes thus leaving p-medians. However, as the problem size increases, the computation time of exact methods increases exponentially. On the other hand, heuristic algorithms generally have acceptable time and memory requirements, but do not guarantee optimal solution. That is, a feasible solution is obtained which is likely to be either optimal or near optimal. One of the first heuristics that was applied to the p-median problem is a greedy heuristic by (Kuehn and Hamburger, 1963), as well heuristic method proposed by Teitz and Bart is one of the oldest and most popular where an interchange algorithm is applied (Teitz and Bart, 1968).

The most popular Metaheuristics appeared in literature are, Genetic Algorithm where the gene of a chromosome consists of indexes of nodes that were selected for the solutions, (Alp et al., 2003). The population size in the introduced algorithm was presented as follows,

$$P(n,p) = \max\left\{2, \left[\frac{n}{100} * \frac{\ln(S)}{d}\right]\right\} d\,, \qquad 2.6$$

where, $S = C\binom{n}{p}$ is number of all possible solutions to the problem and $d = \lfloor n/p \rceil$ the rounded up density to the problem. According to the formula, the result of the max operator is at least two, guaranteeing that every gene appears at least twice in the initial population. This algorithm is simple, fast and generates excellent solutions. (Correa et al., 2006) proposed a genetic algorithm for capacitated p-median problem. (Lorena and Furtado, 2001) offered a genetic algorithm which differs from classical one where a dynamic population and two separate fitness functions are introduced. (Murray and Church, 1996), applied simulated annealing to location-planning models. And more recently (Levanova and Loresh, 2004) implemented the ant systems and simulated annealing algorithm to solve p-median problem. One of the most recent works is related to Variable Neighborhood Search where a new technique is proposed by (Crainic et al 2003, 2004) Cooperative Neighborhood VNS. More information about p-median problem can be found in (Mladenović Nenad et al. 2007).

CHAPTER 3

HEURISTIC ALGORITHMS

3.1. Proposed Discrete Particle Swarm Optimization Algorithm

In proposed Discrete Particle Swarm Optimization algorithm (DPSO), the initial population is generated randomly. Initially, each individual with its position, and fitness value is assigned to its personal best (i.e., the best value of each individual found so far). The best individual in the whole swarm with its position and fitness value, on the other hand, is assigned to the global best (i.e., the best particle in the whole swarm). Then, the position of each particle is updated based on the personal best and the global best. These operations in proposed DPSO are similar to classical PSO algorithm. However, the search strategy of proposed DPSO is different. That is, each particle in the swarm moves based on the following equations.

$$
\begin{aligned}
s_1 &= w^t \oplus \eta(X_i^t) \\
w^{t+1} &= w \cdot \beta \\
s_2 &= c_1 \oplus \eta(P_i^t) \\
s_3 &= c_2 \oplus \eta(G^t) \\
X_i^{t+1} &= best(s_1; s_2; s_3)
\end{aligned}
\qquad 3.1
$$

At each iteration, the position vector of each particle, its personal best and the global best are considered. First of all, a random number of U(0,1) is generated to compare with the inertia weight to decide whether to apply *Exchange* function(η) to the particle or not.

Exchange function (η) implies the exchange of randomly chosen facility with another randomly chosen node. For instance, for the p-median problem of p=4, suppose a sequence of {5, 11, 17, 32} is a possible solution set. In order to apply *Exchange* function, we also need to derive two random numbers; one is for determining the facility to be changed and the other is for the node to be accepted as a new facility. Let's say those numbers are 11 and 20 (that is, the node number 11 will not serve as a facility anymore and the node number 20 will be accepted as a new facility {5, 11, 17, 32}). The new sequence will be {5, 20, 17, 32}. Note that the order of the facilities shown in the solution set is not important. In other words the solution set can be shown in another way too {32, 20, 17, 5}.

If the random number chosen is less than the inertia weight, the particle is manipulated with this *Exchange* function, and the resulting solution, say s_1, is obtained. Meanwhile, the inertia weight is discounted by a constant factor at each iteration, in order to tighten the acceptability of the manipulated particle for the next generation, that is, to diminish the impact of the randomly operated solutions on the swarm evolution.

The next step is to generate another random number of U(0,1) to be compared with c_1, cognitive parameter, to make a decision whether to apply *Exchange* function to personal best of the particle considered. If the random number is less than c_1, then the personal best of the particle undertaken is manipulated and the resulting solution is spared as s_2. Likewise, a third random number of U(0,1) is generated for making a decision whether to manipulate the global best with the *Exchange* function. If the random number is less than c_2, social parameter, then *Exchange* is applied to the global best to obtain a new solution of s_3. Unlike the case of inertia weight, the values of c_1 and c_2 factors are not increased or decreased iteratively, but are fixed at 0.5. That means the probability of applying *Exchange* function to the personal and global bests remains the same. The new replacement solution is selected among s_1, s_2 and s_3, based on their fitness values. This solution may not always be better than the current solution. This is to keep the swarm diverse. The convergence is traced by checking the personal best of each new particle and the global best. As it is seen, proposed equations have all major characteristics of the classical PSO equations. The following pseudo-code describes in detail the steps of the proposed DPSO algorithm.

```
Begin
 Initialize particles (population) randomly
 For each particle
    Calculate fitness value
    Set to position vector and fitness value as personal best (P_i^t)
    Select the best particle and its position vector as global best(G_t)
 End
 Do{
    Update inertia weight
    For each particle
     Apply insert with the probability of inertia weight (s_1)
     Apply insert to (P_i^t) with the probability of c_1 (s_2)
     Apply insert to (G_t) with the probability of c_2 (s_3)
     Select the best one among the s_1,s_2 and s_3
     Update personal best (P_i^t)
    End
    Update global best (G_t)
 }While (Maximum Iteration is not reached)
End
```

Figure 3.1 Pseudo-code for the DPSO algorithm

3.2. Proposed Continuous Particle Swarm Optimization Algorithm

In a proposed novel Continuous Particle Swarm Optimization algorithm (NCPSO), the operations are similar to classical PSO algorithm. However, the initialization of population strategy of NCPSO is different. The position and velocity vectors of initial population are generated randomly and each individual sorted according to its position vector ascendingly. And the first p nodes are opened as facilities to serve the demanding nodes. Initially, each individual with its position, and fitness value is assigned to its personal best (i.e., the best value of each individual found so far). The best individual in the whole swarm with its position and fitness value, on the other hand, is assigned to the global best (i.e., the best particle in the whole swarm). Then, all particles fly across the problem space following the particle that is nearest to the optimum as in the classical PSO algorithm.

Table 3.1: Deriving open facility vector from position

i^{th} Particle Vectors	Particle Dimension (k)				
	1	2	3	4	5
Position Vector(X_i)	1.8	0.72	-0.99	3.01	-5.45
Open Facility Vector (Y_i)	0	1	1	0	1

Below it is shown how sequence is derived from the vector (Y_i) which was derived from the Position Vector (X_i).

Table 3.2: Sequence of facilities opened derived from Open Facilities Vector (Y_i)

i^{th} Particle Vectors	Sequence for p=3		
	1	2	3
Sequence of opened facilities	2	3	5

3.3. Local Search

Local search methods are important part of heuristic optimization which allows us to avoid the local minima. This method starts from some initial solution and iteratively tries to replace the current solution by a better solution in an appropriately defined neighborhood of the current solution. Alternatively, it is called neighborhood search algorithm. First applications of local search have been presented in the late fifties and the early sixties (Croes, 1958), (Lin, 1965).

The proposed local search method in DPSO is applied as, choosing a random facility in a solution set and changing with the demanding nodes which were not selected before in a solution set. However in NCPSO in spite of experimenting various local search methods it did not improve the performance of the algorithm, so we did not use any local search method in a NCPSO.

3.4. Variable Neighborhood Search Algorithm

VNS is one of the most recent metaheuristics developed in combinatorial optimization problem solving in an easier way. It is known as one of very well-known local search methods. VNS gets more attention day-by-day due to its ease of use and accomplishments in solving combinatorial optimization problems.

The VNS is a simple and effective search procedure that proceeds to a systematic change of neighborhood. An ordinary VNS algorithm gets an initial solution $x \in S$, where S is the whole set of search space, then manipulates it through a two nested loop in which the core one alters and explores via two main functions so called shake and local search. The outer loop works as a refresher reiterating the inner loop, while the inner loop carries the major search. Local search explores an improved solution within the local neighborhood, while shake diversifies the solution by switching to another local neighborhood.

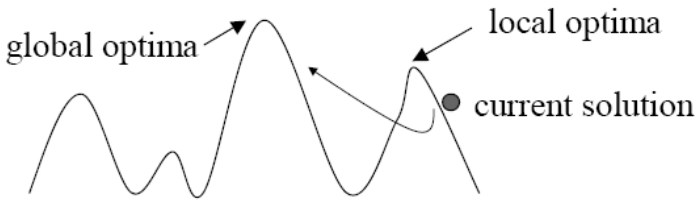

Figure 3.2 Diversification by shake function

The inner loop iterates as long as it keeps improving the solutions, where an integer, *k*, controls the length of the loop. Once an inner loop is completed, the outer loop re-iterates until the termination condition is met. Since the complementariness of neighborhood functions is the key idea behind VNS, the neighborhood structure should be chosen very rigorously in order to achieve an efficient VNS. (Uysal, H. 2006).

In order to develop an effective VNS algorithm, one needs two kinds of neighborhood functions: shake functions ($N_k^s(x)$) and local search functions ($N_l^{LS}(x)$). Each neighborhood function has a particular neighborhood structure. The neighborhood

structures may be used more than one for each function (shake and local search) so as to achieve a valuable neighborhood change. For that purpose, the counters k ($1 \le k \le k_{\max}$) and l ($1 \le l \le l_{\max}$) are used for shake and local search functions respectively in order to ease switching from one to another neighborhood.

```
Begin
 Find an initial solution x
 Do{
    Shake Procedure: Generate at random a starting solution x′ ∈ N_k^S(x).
    Local Search: Apply a local search from the starting solution x′
                  using the base neighborhood structure N_l^LS(x) until
                  a local minimum x″ ∈ N_l^LS(x) is found.
    Improve or not: If x″ is better than x, do x ← x′
 }While (the stopping condition is not met)
End
```

Figure 3.3 Steps of the VNS procedure

The stopping condition may be a maximum CPU time allowed, a maximum number of iterations, or maximum number of iterations between two improvements.

One of the most recent works on Variable Neighborhood Search related to p-median problem where a new technique is proposed, Cooperative Neighborhood VNS (Crainic et al 2003, 2004).

3.4.1. Reduced Variable Neighborhood Search

Reduced VNS has the same analogy as the Basic VNS except that no Local Search procedure is applied. The Reduced VNS explores only randomly different neighborhoods. It can be faster than standard local search algorithms for reaching good quality solutions.

3.5. Simulated Annealing

As its name implies, the Simulated Annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system.

The algorithm is based upon that of (Metropolis et al., 1958) which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was first noted by (Pincus, 1970) but it was (Kirkpatrick et al., 1983) who proposed that it form the basis of an optimization technique for combinatorial (and other) problems.

SA's major advantage over other methods is an ability to avoid becoming trapped at local minima. The algorithm employs a random search which not only accepts changes that decrease objective function **f**, but also some changes that increase it. The latter are accepted with a probability

$$p = \exp\left(-\frac{\delta f}{T}\right) \qquad 3.2$$

where δf is the increase in **f** and **T** is a control parameter, which by analogy with the original application is known as the system “temperature” irrespective of the objective function involved. (Oak Ride National Laboratory, 1996)

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, a comparison study is carried out on the effectiveness of the proposed DPSO and NCPSO algorithms. Proposed DPSO was exclusively tested in comparison with NCPSO algorithm and result of (Domínguez and Muñoz, 2008). The results for DPSO were compared to the simple Simulated Annealing and Reduced Variable Neighborhood Search algorithm as well which were coded according to the pseudo-code in previous chapters. A data set of 40 p-median problems with known optimal solutions in the OR Library (Beasley JE., 1990) was used in testing of performances of algorithms. All proposed DPSO and NCPSO algorithms as well as simple Simulated Annealing and Reduced Variable Neighborhood Search algorithm are coded in C and run on a PC with the configuration of 2.6 GHz CPU and 512MB memory. The size of the population considered by all algorithms is twice the number of nodes.

For DPSO and NCPSO, the social and cognitive parameters were taken as $c_1 = c_2 = 0.5$, initial inertia weight is set to 0.5 and, and the decrement factor β is fixed at 0.9995.

Relative percentile deviation is taken as a performance measure to compare the performance of the algorithms.

$$\frac{f - f_{opt}}{f_{opt}} \cdot 100 \qquad 4.1$$

where f denotes best solution found by the algorithm and f_{opt} denotes the optimum value of the objective function.

4.1. Comparison of DPSO to neural model

Table given below summarizes the comparison of computational results for 10 replications of DPSO and neural model by (Domínguez and Muñoz, 2008) (NA-L+) solving 40 test problems from OR Library. The objective function values found by the algorithms are shown in column 4 and 5. The % error is shown in columns 6 and 7 and the computational time in seconds is shown in the column 8 and 9 respectively.

From the Table given below you can see that the proposed DPSO algorithm performs better not only in terms of the percentile error but it finds the “near optimal” solution in less computational time as well.

For the reason that in the published work some of the results were not presented we were not able to compare the average relative percentile deviation and the average computational time more logically but considering just the once that the authors are presenting in the work (Domínguez and Muñoz, 2008).

The average percentile deviation in the work by Domínguez and Muñoz is 0,48% while in problem solution proposed by our algorithm it is 0,45%.

The same situation is with the computational time where it is 2169,31 seconds in the work by Domínguez and Muñoz while in our work it is 358,62 seconds.

Table 4.1 proposed DPSO -L+ and NA-L+ performances for OR Library test problems

Problem	(n, p)	Optimum	Objective function		% Error		CPU time (s)	
			DPSO-L+	NA-L+	DPSO-L+	NA-L+	DPSO-L+	NA-L+
pmed1	(100, 5)	5819	5819	5819	0	0	0,56	0,27
pmed2	(100, 10)	4093	4099	4093	0,15	0	21,60	1,86
pmed3	(100, 10)	4250	4250	4250	0	0	2,02	0,83
pmed4	(100, 20)	3034	3034	3038	0	0,13	11,84	21,37
pmed5	(100, 33)	1355	1356,5	1359	0,11	0,3	33,79	27,46
pmed6	(200, 5)	7824	7824	*****	0	*****	1,37	*****
pmed7	(200, 10)	5631	5631	5631	0	0	18,72	7,37
pmed8	(200, 20)	4445	4445	4448	0	0,07	21,06	77,5
pmed9	(200, 40)	2734	2740,5	2751	0,24	0,62	119,44	120,89
pmed10	(200, 67)	1255	1256,5	1264	0,12	0,72	105,80	167,07
pmed11	(300, 5)	7696	7696	7696	0	0	2,04	3,23
pmed12	(300, 10)	6634	6634	*****	0	*****	5,13	*****
pmed13	(300, 30)	4374	4374	*****	0	*****	42,93	*****
pmed14	(300, 60)	2968	2972,9	2983	0,17	0,51	144,05	388,51
pmed15	(300, 100)	1729	1733,7	1751	0,27	1,27	150,24	526,11
pmed16	(400, 5)	8162	8162	*****	0	*****	6,36	*****
pmed17	(400, 10)	6999	7000,2	6999	0,02	0	57,98	94,02
pmed18	(400, 40)	4809	4817,5	4811	0,18	0,04	150,48	787,03
pmed19	(400, 80)	2845	2863,7	2863	0,66	0,63	158,84	1024,5
pmed20	(400, 133)	1789	1805,4	1815	0,92	1,45	293,66	1317
pmed21	(500, 5)	9138	9138	*****	0	*****	4,24	*****
pmed22	(500, 10)	8579	8579	*****	0	*****	33,00	*****
pmed23	(500, 50)	4619	4650,6	4624	0,68	0,11	155,87	1889,5
pmed24	(500, 100)	2961	2989,6	2986	0,97	0,84	394,75	2157,2
pmed25	(500, 167)	1828	1845,3	1865	0,95	2,02	727,81	2634,6
pmed26	(600, 5)	9917	9917	*****	0	*****	20,79	*****
pmed27	(600, 10)	8307	8307,9	8307	0,01	0	74,44	171,75
pmed28	(600, 60)	4498	4539,6	4508	0,92	0,22	260,62	3368,7
pmed29	(600, 120)	3033	3062	3060	0,96	0,89	708,85	3827,8
pmed30	(600, 200)	1989	2007,5	2016	0,93	1,36	1972,30	4705,3
pmed31	(700, 5)	10086	10086,1	*****	0	*****	53,87	*****
pmed32	(700, 10)	9297	9297,4	*****	0	*****	77,96	*****
pmed33	(700, 70)	4700	4744	4706	0,94	0,13	451,27	5927,2
pmed34	(700, 140)	3013	3042,2	3038	0,97	0,83	1303,58	6747,6
pmed35	(800, 5)	10400	10400	*****	0	*****	25,38	*****
pmed36	(800, 10)	9934	9945,5	*****	0,12	*****	113,79	*****
pmed37	(800, 80)	5057	5104,7	5071	0,94	0,28	948,08	9243,1
pmed38	(900, 5)	11060	11060	*****	0	*****	29,19	*****
pmed39	(900, 10)	9423	9423	*****	0	*****	111,10	*****
pmed40	(900, 90)	5128	5177,8	5155	0,97	0,53	1393,01	13331

***** The results were not presented in published work

4.2. Comparison of DPSO to NCPSO

And the next presented Table summarizes the comparison of computational results for 10 replications of proposed DPSO to proposed Novel Continuous Particle Swarm Algorithm solving 40 test problems from OR Library. The objective function value found by the algorithms is shown in column 4 and 5. The % error is shown in columns 6 and 7 and the computational time in seconds is shown in the column 8 and 9 respectively.

From the Table given below you can see that the DPSO algorithm performs better than the proposed NCPSO algorithm in terms of the percentile error and it finds the “near optimal” solution in less computational time as well.

The average percentile deviation in the solution method by DPSO it is 0,30% while in the problem solution proposed by NCPSO algorithm is 4,08%.

The same situation is with the computational time where it is 255,20 seconds in the solution by the proposed DPSO algorithm and 1925,46 seconds in the solution method by the proposed CPSO algorithm.

Table 4.2 proposed DPSO -L+ and NCPSO performances for OR Library test problems

Problem	(n, p)	Optimum	Objective function		% Error		CPU time (s)	
			DPSO-L+	NCPSO	DPSO-L+	NCPSO	DPSO-L+	NCPSO
pmed1	(100, 5)	5819	5819	5829	0	0,17	0,56	5,69
pmed2	(100, 10)	4093	4099	4116,3	0,15	0,57	21,60	13,30
pmed3	(100, 10)	4250	4250	4312,3	0	1,47	2,02	13,13
pmed4	(100, 20)	3034	3034	3114,9	0	2,67	11,84	20,63
pmed5	(100, 33)	1355	1356,5	1402,6	0,11	3,51	33,79	24,76
pmed6	(200, 5)	7824	7824	7839,1	0	0,19	1,37	19,43
pmed7	(200, 10)	5631	5631	5746	0	2,04	18,72	48,00
pmed8	(200, 20)	4445	4445	4603,6	0	3,57	21,06	89,17
pmed9	(200, 40)	2734	2740,5	2870,4	0,24	4,99	119,44	114,44
pmed10	(200, 67)	1255	1256,5	1326,9	0,12	5,73	105,80	153,27
pmed11	(300, 5)	7696	7696	7732,6	0	0,48	2,04	88,64
pmed12	(300, 10)	6634	6634	6754,4	0	1,81	5,13	129,86
pmed13	(300, 30)	4374	4374	4598,2	0	5,13	42,93	308,74
pmed14	(300, 60)	2968	2972,9	3139,4	0,17	5,77	144,05	441,22
pmed15	(300, 100)	1729	1733,7	1841,7	0,27	6,52	150,24	476,61
pmed16	(400, 5)	8162	8162	8183,7	0	0,27	6,36	215,99
pmed17	(400, 10)	6999	7000,2	7206,8	0,02	2,97	57,98	258,98
pmed18	(400, 40)	4809	4817,5	5043,6	0,18	4,88	150,48	782,04
pmed19	(400, 80)	2845	2863,7	3033,5	0,66	6,63	158,84	1079,54
pmed20	(400, 133)	1789	1805,4	1929,3	0,92	7,84	293,66	1424,29
pmed21	(500, 5)	9138	9138	9297,8	0	1,75	4,24	344,63
pmed22	(500, 10)	8579	8579	8798,2	0	2,56	33,00	514,93
pmed23	(500, 50)	4619	4650,6	4871,4	0,68	5,46	155,87	1938,42
pmed24	(500, 100)	2961	2989,6	3168,5	0,97	7,01	394,75	2055,73
pmed25	(500, 167)	1828	1845,3	2009,4	0,95	9,92	727,81	2776,58
pmed26	(600, 5)	9917	9917	10006,4	0	0,90	20,79	592,48
pmed27	(600, 10)	8307	8307,9	8554	0,01	2,97	74,44	873,10
pmed28	(600, 60)	4498	4539,6	4732,4	0,92	5,21	260,62	3842,83
pmed29	(600, 120)	3033	3062	3282,7	0,96	8,23	708,85	4589,88
pmed30	(600, 200)	1989	2007,5	2184	0,93	9,80	1972,30	5668,41
pmed31	(700, 5)	10086	10086,1	10153,7	0	0,67	53,87	1005,50
pmed32	(700, 10)	9297	9297,4	9667	0	3,98	77,96	1569,79
pmed33	(700, 70)	4700	4744	4987,3	0,94	6,11	451,27	5869,41
pmed34	(700, 140)	3013	3042,2	3285,5	0,97	9,04	1303,58	7609,70
pmed35	(800, 5)	10400	10400	10543,4	0	1,38	25,38	1608,46
pmed36	(800, 10)	9934	9945,5	10241,3	0,12	3,09	113,79	2790,84
pmed37	(800, 80)	5057	5104,7	5383,571	0,94	6,46	948,08	9838,69
pmed38	(900, 5)	11060	11060	11173,2	0	1,02	29,19	2170,54
pmed39	(900, 10)	9423	9423	9790,3	0	3,90	111,10	2528,09
pmed40	(900, 90)	5128	5177,8	5469,333	0,97	6,66	1393,01	13122,63

4.3. Comparison of DPSO to RVNS for 1000 generations

Table given below summarizes the comparison of computational results for 10 replications of DPSO and Reduced Variable Neighborhood Search Algorithm solving 40 test problems from OR Library. The smallest and the biggest percentile error made for the RVNS is shown in column 4 and 5. The smallest and the biggest percentile error made for the DPSO is shown in column 6 and 7 and the computational time in seconds is shown in the column 8 and 9 respectively.

From the Table given below you can see that the proposed DPSO algorithm performs better not only in terms of the percentile error but it finds the “near optimal” solution in less computational time for some of the problems.

The average percentile deviation in RVNS is 8,84% while in problem solution proposed by our algorithm it is 0,30%. These values are not shown in the table.

In the computational results shown in table below RVNS was run 1000 generations. The generation number was selected in a such way in order to compare the results after running the algorithms under the same conditions. In most cases it resulted in less computational time compared to DPSO.

Table 4.3 Comparison of DPSO to RVNS for 1000 generations

Problem	(n, p)	Optimum	Error%				CPU time (s)	
			RVNS		DPSO		RVNS	DPSO
			Best	Worst	Best	Worst		
pmed1	(100, 5)	5819	0	0,60	0	0	23,05	0,56
pmed2	(100, 10)	4093	0,17	2,71	0	0,29	25,87	21,6
pmed3	(100, 10)	4250	0	3,67	0	0	25,58	2,02
pmed4	(100, 20)	3034	1,45	6,43	0	0	25,92	11,84
pmed5	(100, 33)	1355	2,51	8,63	0	0,22	25,95	33,79
pmed6	(200, 5)	7824	0	0,55	0	0	23,09	1,37
pmed7	(200, 10)	5631	0,66	3,46	0	0	26,00	18,72
pmed8	(200, 20)	4445	2,72	5,96	0	0	26,08	21,06
pmed9	(200, 40)	2734	3,44	7,50	0	0,62	26,19	119,44
pmed10	(200, 67)	1255	3,35	10,12	0	0,40	26,34	105,8
pmed11	(300, 5)	7696	0	1,53	0	0	25,38	2,04
pmed12	(300, 10)	6634	1,39	4,16	0	0	26,16	5,13
pmed13	(300, 30)	4374	2,95	6,93	0	0	26,38	42,93
pmed14	(300, 60)	2968	6,10	8,66	0	0,47	26,63	144,05
pmed15	(300, 100)	1729	6,54	11,57	0,06	0,52	26,92	150,24
pmed16	(400, 5)	8162	0,01	2,28	0	0	26,19	6,36
pmed17	(400, 10)	6999	1,70	5,33	0	0,06	26,34	57,98
pmed18	(400, 40)	4809	3,78	6,49	0,04	0,48	26,82	150,48
pmed19	(400, 80)	2845	6,15	9,00	0,35	0,98	27,22	158,84
pmed20	(400, 133)	1789	9,39	13,47	0,84	0,95	27,71	293,66
pmed21	(500, 5)	9138	0	3,30	0	0	25,50	4,24
pmed22	(500, 10)	8579	1,08	4,57	0	0	26,53	33
pmed23	(500, 50)	4619	5,59	7,64	0,26	0,97	27,09	155,87
pmed24	(500, 100)	2961	7,16	10,37	0,91	0,98	27,70	394,75
pmed25	(500, 167)	1828	9,52	13,02	0,82	0,98	28,50	727,81
pmed26	(600, 5)	9917	0,30	2,86	0	0	26,15	20,79
pmed27	(600, 10)	8307	1,66	4,78	0	0,04	26,43	74,44
pmed28	(600, 60)	4498	6,07	8,14	0,78	0,98	27,62	260,62
pmed29	(600, 120)	3033	8,90	11,61	0,86	0,99	28,43	708,85
pmed30	(600, 200)	1989	11,31	14,88	0,80	0,96	29,59	1972,3
pmed31	(700, 5)	10086	0,35	3,54	0	0,01	26,52	53,87
pmed32	(700, 10)	9297	2,07	5,16	0	0,04	26,38	77,96
pmed33	(700, 70)	4700	5,47	8,57	0,83	0,98	26,81	451,27
pmed34	(700, 140)	3013	10,36	13,84	0,93	1,00	27,23	1303,58
pmed35	(800, 5)	10400	0,01	3,90	0	0	26,02	25,38
pmed36	(800, 10)	9934	1,80	4,02	0	0,31	26,50	113,79
pmed37	(800, 80)	5057	7,10	9,55	0,83	0,99	27,15	948,08
pmed38	(900, 5)	11060	0,80	2,88	0	0	26,08	29,19
pmed39	(900, 10)	9423	1,89	4,75	0	0	26,32	111,1
pmed40	(900, 90)	5128	7,61	10,04	0,90	0,99	27,39	1393,01
Average			3,53	6,66	0,23	0,38	26,49	255,20

4.4. Comparison of DPSO to RVNS for 5000 generations

Table given below summarizes the comparison of computational results for 10 replications of DPSO and Reduced Variable Neighborhood Search Algorithm solving 40 test problems from OR Library. The smallest and the biggest percentile error made for the RVNS is shown in column 4 and 5. The smallest and the biggest percentile error made for the DPSO is shown in column 6 and 7 and the computational time in seconds is shown in the column 8 and 9 respectively.

From the Table given below you can see that the proposed DPSO algorithm performs better not only in terms of the percentile error but it finds the “near optimal” solution in less computational time for some of the problems for sure.

The average percentile deviation in RVNS is 2,1% while in problem solution proposed by our algorithm it is 0,3%.

In average computational time it seems like the RVNS performs better but we should not forget that in most of the solved problems the RVNS performance was far away compared to DPSO in terms of percentile error. And in most of the cases the percentile error for DPSO is better. The RVNS was run 5000 generations in this case. Compared to 1000 generations run the computational time is more “near” to the computational time in DPSO.

Table 4.4 Comparison of DPSO to RVNS for 5000 generations

Problem	(n, p)	Optimum	Error%				CPU time (s)	
			RVNS		DPSO		RVNS	DPSO
			Best	Worst	Best	Worst		
pmed1	(100, 5)	5819	0	0,00	0	0	30,79	0,56
pmed2	(100, 10)	4093	0	0,29	0	0,29	95,22	21,6
pmed3	(100, 10)	4250	0	0,87	0	0	94,65	2,02
pmed4	(100, 20)	3034	0	0,76	0	0	115,09	11,84
pmed5	(100, 33)	1355	0	1,48	0	0,22	129,11	33,79
pmed6	(200, 5)	7824	0	0,00	0	0	40,01	1,37
pmed7	(200, 10)	5631	0	1,17	0	0	121,48	18,72
pmed8	(200, 20)	4445	0,20	2,92	0	0	130,10	21,06
pmed9	(200, 40)	2734	0,51	2,41	0	0,62	130,48	119,44
pmed10	(200, 67)	1255	1,12	4,46	0	0,40	131,14	105,8
pmed11	(300, 5)	7696	0	0,35	0	0	74,76	2,04
pmed12	(300, 10)	6634	0	1,06	0	0	121,16	5,13
pmed13	(300, 30)	4374	1,07	3,77	0	0	131,21	42,93
pmed14	(300, 60)	2968	2,32	5,26	0	0,47	132,26	144,05
pmed15	(300, 100)	1729	1,74	5,09	0,06	0,52	133,57	150,24
pmed16	(400, 5)	8162	0	0,26	0	0	114,06	6,36
pmed17	(400, 10)	6999	0	2,71	0	0,06	129,87	57,98
pmed18	(400, 40)	4809	1,35	3,10	0,04	0,48	132,84	150,48
pmed19	(400, 80)	2845	2,81	5,27	0,35	0,98	134,92	158,84
pmed20	(400, 133)	1789	4,42	7,55	0,84	0,95	137,12	293,66
pmed21	(500, 5)	9138	0	0,45	0	0	89,04	4,24
pmed22	(500, 10)	8579	0,05	2,07	0	0	132,22	33
pmed23	(500, 50)	4619	2,66	5,82	0,26	0,97	135,53	155,87
pmed24	(500, 100)	2961	3,48	6,72	0,91	0,98	138,45	394,75
pmed25	(500, 167)	1828	4,21	7,06	0,82	0,98	141,85	727,81
pmed26	(600, 5)	9917	0	0,85	0	0	94,33	20,79
pmed27	(600, 10)	8307	0,46	1,66	0	0,04	133,05	74,44
pmed28	(600, 60)	4498	3,07	4,54	0,78	0,98	138,72	260,62
pmed29	(600, 120)	3033	5,31	6,26	0,86	0,99	142,38	708,85
pmed30	(600, 200)	1989	4,52	6,99	0,80	0,96	147,67	1972,3
pmed31	(700, 5)	10086	0	0,74	0	0,01	117,65	53,87
pmed32	(700, 10)	9297	0,26	1,23	0	0,04	132,54	77,96
pmed33	(700, 70)	4700	3,11	4,96	0,83	0,98	140,00	451,27
pmed34	(700, 140)	3013	4,88	6,77	0,93	1,00	145,14	1303,58
pmed35	(800, 5)	10400	0	1,02	0	0	118,00	25,38
pmed36	(800, 10)	9934	0,32	1,41	0	0,31	132,99	113,79
pmed37	(800, 80)	5057	2,79	5,99	0,83	0,99	142,65	948,08
pmed38	(900, 5)	11060	0	0,90	0	0	130,68	29,19
pmed39	(900, 10)	9423	0,33	2,10	0	0	133,32	111,1
pmed40	(900, 90)	5128	3,69	5,21	0,90	0,99	145,89	1393,01
Average			1,37	3,04	0,23	0,38	122,30	255,20

4.5. Comparison of DPSO to Simulated Annealing for 1000 generations

Table given below summarizes the comparison of computational results for 10 replications of DPSO and Simulated Annealing (SA) solving 40 test problems from OR Library. The smallest and the biggest percentile error made for the SA is shown in column 4 and 5. The smallest and the biggest percentile error made for the DPSO is shown in column 6 and 7 and the computational time in seconds is shown in the column 8 and 9 respectively.

From the Table given below you can see that the proposed DPSO algorithm performs better not only in terms of the percentile error but it finds the “near optimal” solution in less computational time for some of the problems.

The average percentile deviation in SA is 6,01% while in problem solution proposed by our algorithm it is 0,30%. These values are not shown in the table.

In the computational results shown in table below SA was run 1000 generations. The generation number was selected in a such way in order to compare the results after running the algorithms under the same conditions. In most cases it resulted in less computational time compared to DPSO.

Table 4.5 Comparison of DPSO to Simulated Annealing for 1000 generations

Problem	(n, p)	Optimum	Error%				CPU time (s)	
			SA		DPSO		SA	DPSO
			Best	Worst	Best	Worst		
pmed1	(100, 5)	5819	0	1,12	0	0	21,79	0,56
pmed2	(100, 10)	4093	0,29	2,96	0	0,29	26,05	21,6
pmed3	(100, 10)	4250	0	3,67	0	0	24,71	2,02
pmed4	(100, 20)	3034	0,96	6,36	0	0	25,89	11,84
pmed5	(100, 33)	1355	3,03	8,56	0	0,22	25,86	33,79
pmed6	(200, 5)	7824	0,12	1,94	0	0	25,86	1,37
pmed7	(200, 10)	5631	0,64	4,49	0	0	25,93	18,72
pmed8	(200, 20)	4445	3,24	6,57	0	0	25,98	21,06
pmed9	(200, 40)	2734	3,69	7,72	0	0,62	26,03	119,44
pmed10	(200, 67)	1255	5,82	15,14	0	0,40	26,08	105,8
pmed11	(300, 5)	7696	0,35	2,29	0	0	25,96	2,04
pmed12	(300, 10)	6634	1,45	6,21	0	0	26,05	5,13
pmed13	(300, 30)	4374	3,06	7,84	0	0	26,18	42,93
pmed14	(300, 60)	2968	6,77	8,96	0	0,47	26,29	144,05
pmed15	(300, 100)	1729	7,98	14,23	0,06	0,52	26,41	150,24
pmed16	(400, 5)	8162	0,29	2,96	0	0	26,10	6,36
pmed17	(400, 10)	6999	1,19	4,91	0	0,06	26,21	57,98
pmed18	(400, 40)	4809	4,53	7,49	0,04	0,48	26,42	150,48
pmed19	(400, 80)	2845	7,21	11,63	0,35	0,98	26,57	158,84
pmed20	(400, 133)	1789	10,96	22,30	0,84	0,95	26,78	293,66
pmed21	(500, 5)	9138	1,26	5,37	0	0	26,31	4,24
pmed22	(500, 10)	8579	2,30	5,47	0	0	26,24	33
pmed23	(500, 50)	4619	5,72	9,37	0,26	0,97	26,44	155,87
pmed24	(500, 100)	2961	8,48	13,41	0,91	0,98	26,63	394,75
pmed25	(500, 167)	1828	13,02	19,58	0,82	0,98	26,95	727,81
pmed26	(600, 5)	9917	1,01	4,13	0	0	26,03	20,79
pmed27	(600, 10)	8307	2,19	5,51	0	0,04	26,21	74,44
pmed28	(600, 60)	4498	5,96	9,52	0,78	0,98	26,68	260,62
pmed29	(600, 120)	3033	9,89	16,16	0,86	0,99	27,00	708,85
pmed30	(600, 200)	1989	13,88	21,72	0,80	0,96	27,43	1972,3
pmed31	(700, 5)	10086	0,20	3,39	0	0,01	26,52	53,87
pmed32	(700, 10)	9297	2,59	4,88	0	0,04	26,38	77,96
pmed33	(700, 70)	4700	8,11	13,06	0,83	0,98	26,81	451,27
pmed34	(700, 140)	3013	12,58	18,39	0,93	1,00	27,23	1303,58
pmed35	(800, 5)	10400	0,53	4,06	0	0	26,02	25,38
pmed36	(800, 10)	9934	2,18	4,59	0	0,31	26,50	113,79
pmed37	(800, 80)	5057	7,53	11,94	0,83	0,99	27,15	948,08
pmed38	(900, 5)	11060	0,37	2,73	0	0	26,08	29,19
pmed39	(900, 10)	9423	3,41	6,02	0	0	26,32	111,1
pmed40	(900, 90)	5128	8,39	11,91	0,90	0,99	27,39	1393,01
Average			4,28	8,46	0,23	0,38	26,24	255,20

4.6. Comparison of DPSO to Simulated Annealing for 5000 generations

Table given below summarizes the comparison of computational results for 10 replications of DPSO and Simulated Annealing (SA) solving 40 test problems from OR Library. The smallest and the biggest percentile error made for the SA is shown in column 4 and 5. The smallest and the biggest percentile error made for the DPSO is shown in column 6 and 7 and the computational time in seconds is shown in the column 8 and 9 respectively.

From the Table given below you can see that the proposed DPSO algorithm performs better not only in terms of the percentile error but it finds the “near optimal” solution in less computational time for some of the problems.

The average percentile deviation in SA is 2,36% while in problem solution proposed by our algorithm it is 0,3%. These values are not shown in the given table.

In average computational time it seems like the SA performs better but we should not forget that in most of the solved problems the SA performance was far away compared to DPSO in terms of percentile error. And in most of the cases the percentile error for DPSO is better. The SA was run 5000 generations in this case. Compared to 1000 generations run the computational time is more “near” to the computational time in DPSO.

Table 4.6 Comparison of DPSO to Simulated Annealing for 5000 generations

Problem	(n, p)	Optimum	Error%				CPU time (s)	
			SA		DPSO		SA	DPSO
			Best	Worst	Best	Worst		
pmed1	(100, 5)	5819	0	0	0	0	29,07	0,56
pmed2	(100, 10)	4093	0	0,29	0	0,29	97,96	21,6
pmed3	(100, 10)	4250	0	0,47	0	0	92,20	2,02
pmed4	(100, 20)	3034	0	1,52	0	0	128,31	11,84
pmed5	(100, 33)	1355	0	2,44	0	0,22	126,85	33,79
pmed6	(200, 5)	7824	0	0	0	0	56,29	1,37
pmed7	(200, 10)	5631	0	0,55	0	0	113,30	18,72
pmed8	(200, 20)	4445	0,13	2,34	0	0	129,53	21,06
pmed9	(200, 40)	2734	0,80	4,24	0	0,62	129,77	119,44
pmed10	(200, 67)	1255	2,07	5,98	0	0,40	130,04	105,8
pmed11	(300, 5)	7696	0	0,44	0	0	75,53	2,04
pmed12	(300, 10)	6634	0	0,69	0	0	120,85	5,13
pmed13	(300, 30)	4374	1,07	3,54	0	0	130,30	42,93
pmed14	(300, 60)	2968	2,80	5,76	0	0,47	130,67	144,05
pmed15	(300, 100)	1729	3,53	6,07	0,06	0,52	131,25	150,24
pmed16	(400, 5)	8162	0	0,34	0	0	125,23	6,36
pmed17	(400, 10)	6999	0,06	1,74	0	0,06	130,23	57,98
pmed18	(400, 40)	4809	1,89	4,70	0,04	0,48	131,12	150,48
pmed19	(400, 80)	2845	3,90	6,50	0,35	0,98	131,87	158,84
pmed20	(400, 133)	1789	4,92	8,27	0,84	0,95	132,89	293,66
pmed21	(500, 5)	9138	0	1,01	0	0	116,59	4,24
pmed22	(500, 10)	8579	0,15	2,02	0	0	131,72	33
pmed23	(500, 50)	4619	3,27	4,87	0,26	0,97	133,21	155,87
pmed24	(500, 100)	2961	3,68	5,81	0,91	0,98	134,27	394,75
pmed25	(500, 167)	1828	5,14	8,75	0,82	0,98	135,78	727,81
pmed26	(600, 5)	9917	0	1,03	0	0	116,48	20,79
pmed27	(600, 10)	8307	0,42	1,78	0	0,04	131,35	74,44
pmed28	(600, 60)	4498	2,78	4,82	0,78	0,98	133,67	260,62
pmed29	(600, 120)	3033	4,88	6,96	0,86	0,99	135,14	708,85
pmed30	(600, 200)	1989	5,53	7,44	0,80	0,96	137,30	1972,3
pmed31	(700, 5)	10086	0	0,77	0	0,01	124,90	53,87
pmed32	(700, 10)	9297	0,41	2,08	0	0,04	131,46	77,96
pmed33	(700, 70)	4700	3,11	5,57	0,83	0,98	134,29	451,27
pmed34	(700, 140)	3013	5,34	7,04	0,93	1,00	136,25	1303,58
pmed35	(800, 5)	10400	0	1,62	0	0	117,19	25,38
pmed36	(800, 10)	9934	0,49	2,32	0	0,31	131,80	113,79
pmed37	(800, 80)	5057	3,94	5,14	0,83	0,99	135,47	948,08
pmed38	(900, 5)	11060	0	0,84	0	0	129,83	29,19
pmed39	(900, 10)	9423	0,36	1,51	0	0	131,37	111,1
pmed40	(900, 90)	5128	3,92	5,50	0,90	0,99	136,69	1393,01
Average			1,62	3,32	0,23	0,38	122,20	255,20

CHAPTER 5

CONCLUSION

The main objective in this work is to allocate p facilities in a way that the total distance between *n* demand points and facilities minimized. This is a well know facility-location problem and in literature is known as a *p*-median problem.

It is shown by Kariv and Hakimi that the p-median problem is NP-hard (Kariv and Hakimi, 1979b). It is unlikely to obtain optimal schedule through polynomial time-bounded algorithms. Small size instances of p-median problem can be solved with reasonable computational time by exact algorithms such as branch-and-bound (Järvinen et al., 1972).

On the other hand, heuristic algorithms generally have acceptable time and memory requirements, but do not guarantee optimal solution. That is, a feasible solution is obtained which is likely to be either optimal or near optimal.

Particle swarm optimization (PSO) is one of the latest metaheuristic methods in literature, which is based on the metaphor of social interaction and communication among different spaces in nature, such as bird flocking and fish schooling. PSO was first introduced to optimize various continuous nonlinear functions by (Eberhart and Kennedy, 1995).

p-Median problem is related to the Discrete Location Problems while the Particle swarm optimization (PSO) is manly designed for the continuous problems. Thus it may have some drawbacks when applying PSO to discrete problems. Recently a few researches

have been related to the discrete combinatorial optimization problems. However, still it is considered that the applications of PSO on discrete problems are limited.

In this thesis Novel proposed Continuous Particle Swarm (NCPSO) algorithm and a proposed Discrete Particle Swarm Optimization algorithms (DPSO) are proposed. The algorithms have been tested on benchmark problem instances from OR Library and compared to the other algorithms in literature and shown that the proposed algorithm results in better computational time. To the best of our knowledge the proposed PSO algorithms in this thesis are the first PSO algorithms applied to the p-median problem.

REFERENCES

Afshinmanesh, F., Marandi, A., and Rahimi-Kian, A., *A novel binary particle swarm optimization method using artificial immune system*, International Conference on Computer as a Tool, EUROCON, pp.217–220, 2005

Allahverdi, A., Al-Anzi, F.S., *Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times*, International Journal of Production Research, 44(22), 4713–4735, 2006.

Alp, O., Erkut, E. and Drezner Z., *An efficient genetic algorithm for the p-median problem,* Annals of Operations Research, 122:21–42, 2003.

Beasley JE. OR-Library: "*distributing test problems by electronic mail*", Journal of the Operational Research Society, 41(11), 1069–72, 1990.

Daskin M.S., *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley and Sons, Inc., New York, 1995.

Camci, F., *Comparison of genetic and binary particle swarm optimization algorithms on system maintenance scheduling using prognostics information*, Engineering Optimization,41:2, pp.119 -136, 2009

Cornuéjols G., Nemhauser G. L., and Wolsey L. A., *The uncapacitated facility location problem, in* Discrete *Location Theory*, pp. 119–171, John Wiley & Sons, New York, NY, USA, 1990.

Correa, E. S. Steiner, M. T. A. Freitas, A. A. and Carnieri, C., *A genetic algorithm for solving a capacitated p-median problem.* Numerical Algorithms, 35:373–388, 2004

Crainic, T. G. Gendreau, M. Hansen, P. and Mladenovi´c, N. *Parallel variable neighborhood search for the p-median*. Les Cahiers du GERAD, G-2003-4, 2003

Crainic, T. G. Gendreau, M. Hansen, P. and N. Mladenovi´c. *Cooperative parallel variable neighborhood search for the p-median.* Journal of Heuristics, 10(3):293–314. 2004

Croes, G.A., *A Method for Solving Traveling-Salesman Problems*, Operations Research 6, 791-812, 1958

Domínguez E. and Muñoz, J., *A neural model for the p-median problem.* Computers & Operations Research 35, 404 – 416, 2008.

Eberhart, R.C., and Kennedy, J., *A new optimizer using particle swarm theory,* Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39-43, 1995.

Guner A.R., Sevkli M., *A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem*, Journal of Artificial Evolution and Applications. No. 10, pp. 1687-6229, 2008.

Järvinen, P., Rajala, J. and Sinervo, H. *A branch-and-bound algorithm for seeking the p-median*, Operations Research, 20(1): pp.173–178, 1972.

Kariv, O., Hakimi, S.L., *An algorithmic approach to network location problems. Part 1. The p-centers*. SIAM Journal on Applied Mathematics 37, 513–538, 1979.

Kariv, O., Hakimi, S. L., *An algorithmic approach to network location problems. II. The p-medians.* SIAM Journal on Applied Mathematics, 37(3):539–560, 1979

Kennedy, J. and Eberhart, R.C., *A discrete binary version of the particle swarm algorithm.* Proceedings of the International Conference on Systems, Man, Cybernetics. Piscataway, NJ, 4104–4109, 1997.

Kennedy, J., Eberhart, R.C. and Shi, Y. *Swarm Intelligence*, San Mateo, Morgan Kaufmann, CA, USA, 2001.

Kirkpatrick, S., Gerlatt, C. D. Jr., and Vecchi, M.P., *Optimization by Simulated Annealing*, Science 220, 671-680, 1983

Kuehn, A.A. and Hamburger, M.J.. *A heuristic program for locating warehouses.* Management Science, 9(4):643–666, 1963

Levanova, T. V. and Loresh, M. A. *Algorithms of ant system and simulated annealing for the p-median problem.* Automation and Remote Control, 65(3):431–438, 2004.

Lin, S., *Computer Solutions of the Travelling Salesman Problem*, Bell Systems Technical Journal 44, 2245–2269, 1965

Lorena, L. A. N. and Furtado, J. C.. *Constructive genetic algorithm for clustering problems.* Evolutionary Computation, 9(3):309–328, 2001.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., Teller, A.H. and Teller, E., *Equations of State Calculations by Fast Computing Machines*, J. Chem. Phys. 21, 1087-1092, 1958.

Mirchandani P.B., Francis R.L., *Discrete Location Theory*, John Wiley & Sons, 1990.

Mladenović Nenad, Brimberg Jack, Hansen Pierre, Moreno-Pérez Jose A., *The p-median problem: A survey of metaheuristic approaches.* European Journal of Operational Research 179: 927–939, 2007.

Murray A.T, Church R.L. *Applying simulated annealing to location-planning models.* Journal of Heuristics, 2:31–53. 1996.

Ride National Laboratory, http://www.phy.ornl.gov, 1996

Onwubolu, G.C. and M. Clerc. *Optimal Operating Path for Automated Drilling Operations by a New Heuristic Approach Using Particle Swarm Optimization*. International Journal of Production Research 42(3), 473-491, 2004

Pan, Q-K., Tasgetiren, M.F., and Liang,Y-C, *A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem*, Computers & Operations Research, 35(9), 2807-2839, 2008.

Pincus, M., *A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems*, Oper. Res. 18, 1225-1228, 1970

ReVelle, C. and R. Swain. *Central Facilities Location.* Geographical Analysis 2, 30–42, (1970).

Sobolev Institute of Mathematics, http://www.math.nsc.ru/AP/benchmarks/english.html, 2009

Sridharan R., *The capacitated plant location problem.* European Journal of Operational Research. v. 87 1995, *pp*. 203–213, 1995

Tasgetiren M.F, Liang, Y-C, Sevkli, M., Gencyilmaz, G. *Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem,* International Journal of Production Research, 44(22), 4737–4754, 2006

Tasgetiren, M.F., Liang, Y-C., Sevkli, M. and Gencyilmaz, G, Particle *Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in Permutation Flowshop Sequencing Problem*, European Journal of Operational Research 177 (3), 1930-1947, 2007.

Teitz, M. B. and Bart, P., *Heuristic methods for estimating the generalized vertex median of a weighted graph,* Operations Research, 16(5):955–961, 1968.

Tseng, C-T. and Liao, C-J. *A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks*, International Journal of Production Research, 46(17), 4655–4670, 2008.

Uysal, H., *A variable neighborhood search algorithm for identical parallel machine problem*, M.S. Thesis, Fatih University, 2006.

Van den Bergh, F. and A.P. Engelbecht. *Cooperative Learning in Neural Networks Using Particle Swarm Optimizers.* South African Computer Journal 26, 84-90, 2000

APPENDIX A

DATASET

Here shown 5 example problems of dataset used from OR Library (Beasley, 1990) for testing the proposed algorithms.

Data are shown in the following format:

Two nodes are given a and b separated by sign "–"the distance between two nodes is shown just after them.

pmed1:		**n-100**		**edge-200**		**optimal-5819**			
1-2	30	2-3	46	3-4	1	4-5	28	5-6	31
6-7	69	7-8	39	8-9	14	9-10	84	10-11	59
11-12	10	12-13	28	13-14	63	14-15	9	15-16	100
16-17	98	17-18	70	18-19	94	19-20	22	20-21	14
21-22	87	22-23	82	23-24	55	24-25	2	25-26	32
26-27	77	27-28	95	28-29	29	29-30	59	30-31	91
31-32	89	32-33	50	33-34	40	34-35	88	35-36	94
36-37	60	37-38	21	38-39	89	39-40	47	40-41	63
41-42	45	42-43	46	43-44	24	44-45	77	45-46	60
46-47	45	47-48	50	48-49	93	49-50	22	50-51	84
51-52	16	52-53	85	53-54	68	54-55	93	55-56	37
56-57	26	57-58	29	58-59	38	59-60	10	60-61	32
61-62	67	62-63	66	63-64	52	64-65	19	65-66	39
66-67	12	67-68	86	68-69	72	69-70	73	70-71	65
71-72	2	72-73	8	73-74	96	74-75	43	75-76	39
76-77	61	77-78	90	78-79	8	79-80	58	80-81	91
81-82	58	82-83	13	83-84	79	84-85	59	85-86	28
86-87	46	87-88	24	88-89	63	89-90	81	90-91	14
91-92	52	92-93	64	93-94	75	94-95	71	95-96	51
96-97	75	97-98	57	98-99	31	99-100	49	100-1	88
80-5	90	85-69	51	20-19	30	85-13	22	81-95	54
58-73	37	64-51	89	39-76	89	13-8	77	91-76	51
52-95	96	91-75	33	80-91	84	35-80	46	54-86	35
30-70	5	19-7	9	74-53	50	11-37	4	19-92	70

62-94	96	60-52	34	24-59	63	42-45	68	19-60	53
60-27	86	15-50	31	38-18	27	81-30	99	1-29	6
22-52	11	54-59	75	5-7	8	41-44	80	41-76	42
44-12	56	61-65	73	20-93	35	9-99	28	45-68	19
16-79	96	13-42	3	68-4	28	58-17	60	21-81	34
35-7	7	21-14	82	26-32	26	6-63	83	31-69	53
91-78	32	45-59	33	42-57	5	8-53	66	65-94	61
94-90	63	73-33	88	39-21	59	42-85	34	30-57	37
34-94	52	62-38	56	26-29	20	48-98	53	4-33	45
31-63	82	8-52	31	37-81	14	50-27	16	27-14	48
88-33	8	56-30	76	87-91	17	33-49	77	70-30	74
25-99	19	95-30	96	14-85	90	49-8	96	96-5	87
73-4	41	99-32	35	95-1	31	99-3	49	18-75	100
58-47	68	35-60	5	38-16	70	54-40	52	95-55	47
54-17	45	9-83	75	97-66	81	9-48	92	90-95	96
4-27	27	75-22	48	62-7	55	79-3	57	15-69	46

pmed2:		**n-100**		**edge-200**		**optimal-4093**			
1-2	24	2-3	49	3-4	72	4-5	46	5-6	22
6-7	86	7-8	7	8-9	52	9-10	48	10-11	72
11-12	10	12-13	33	13-14	50	14-15	98	15-16	96
16-17	39	17-18	89	18-19	80	19-20	77	20-21	3
21-22	63	22-23	45	23-24	8	24-25	84	25-26	22
26-27	9	27-28	71	28-29	99	29-30	77	30-31	65
31-32	71	32-33	18	33-34	81	34-35	67	35-36	11
36-37	21	37-38	85	38-39	62	39-40	36	40-41	1
41-42	52	42-43	100	43-44	25	44-45	43	45-46	61
46-47	41	47-48	72	48-49	93	49-50	45	50-51	100
51-52	27	52-53	27	53-54	76	54-55	49	55-56	71
56-57	24	57-58	8	58-59	91	59-60	81	60-61	100
61-62	83	62-63	27	63-64	58	64-65	98	65-66	50
66-67	5	67-68	26	68-69	94	69-70	100	70-71	19
71-72	58	72-73	27	73-74	94	74-75	64	75-76	48
76-77	19	77-78	83	78-79	87	79-80	94	80-81	41
81-82	1	82-83	67	83-84	37	84-85	69	85-86	52
86-87	83	87-88	6	88-89	78	89-90	1	90-91	57
91-92	60	92-93	22	93-94	99	94-95	25	95-96	11
96-97	11	97-98	54	98-99	69	99-100	7	100-1	94
2-83	2	35-61	56	32-95	37	70-58	18	14-73	96
97-55	36	49-54	84	37-1	39	22-77	16	3-74	98
28-60	77	11-14	94	2-36	47	20-58	42	3-25	50
83-96	40	97-29	87	52-33	41	37-72	38	32-44	53
68-14	80	60-18	54	45-78	56	7-86	36	55-23	57
89-88	39	37-18	80	83-48	43	56-3	30	19-50	94
54-99	11	88-50	46	36-16	14	22-77	17	37-16	12
80-79	56	84-96	42	69-89	64	84-37	40	21-22	79
65-23	45	22-83	95	46-94	75	14-21	50	84-44	41

64-71	15	62-84	97	67-30	8	34-38	56	58-91	2
57-90	98	79-88	80	96-5	40	61-94	99	98-79	77
5-51	80	28-41	64	18-47	78	53-97	76	67-1	48
23-71	9	52-41	25	11-50	94	18-24	26	6-27	5
18-53	71	40-73	48	51-64	42	50-72	4	18-86	85
91-2	7	12-63	23	2-56	85	19-36	14	53-24	68
73-40	37	93-6	5	98-73	3	51-43	57	63-11	79
85-51	27	10-31	30	42-4	15	19-65	79	98-20	32
99-14	91	67-30	15	8-55	46	91-12	63	10-33	54
72-48	93	64-30	83	23-37	2	87-16	21	81-73	86
44-26	55	81-12	32	54-26	82	79-88	61	34-67	63

pmed3:		**n-100**		**edge-200**		**optimal-4250**			
1-2	77	2-3	62	3-4	90	4-5	22	5-6	17
6-7	19	7-8	11	8-9	4	9-10	3	10-11	16
11-12	67	12-13	35	13-14	21	14-15	14	15-16	43
16-17	11	17-18	91	18-19	55	19-20	33	20-21	80
21-22	90	22-23	81	23-24	91	24-25	23	25-26	49
26-27	62	27-28	96	28-29	25	29-30	92	30-31	77
31-32	41	32-33	40	33-34	97	34-35	1	35-36	14
36-37	18	37-38	69	38-39	72	39-40	17	40-41	56
41-42	94	42-43	78	43-44	25	44-45	99	45-46	51
46-47	24	47-48	33	48-49	47	49-50	87	50-51	71
51-52	88	52-53	60	53-54	96	54-55	10	55-56	34
56-57	56	57-58	56	58-59	99	59-60	35	60-61	86
61-62	85	62-63	18	63-64	29	64-65	48	65-66	82
66-67	43	67-68	59	68-69	8	69-70	10	70-71	58
71-72	62	72-73	49	73-74	94	74-75	84	75-76	69
76-77	5	77-78	36	78-79	17	79-80	47	80-81	88
81-82	39	82-83	84	83-84	56	84-85	82	85-86	98
86-87	93	87-88	91	88-89	73	89-90	65	90-91	26
91-92	73	92-93	19	93-94	30	94-95	76	95-96	31
96-97	37	97-98	46	98-99	9	99-100	39	100-1	66
65-66	39	61-6	96	15-81	44	95-43	37	13-57	53
78-7	7	32-26	15	14-66	49	25-44	42	80-84	33
79-58	99	25-59	38	4-37	32	1-51	49	54-20	5
39-18	87	74-8	71	69-3	64	13-30	38	1-66	62
48-72	30	33-31	57	79-94	79	35-92	54	77-69	28
41-49	88	82-44	18	36-24	6	47-41	26	4-31	53
84-56	42	2-37	75	43-3	55	58-57	3	26-42	51
80-62	32	50-70	98	55-60	87	9-77	10	45-23	73
39-83	95	1-35	71	80-26	82	81-26	57	91-37	89
83-42	53	21-84	33	45-21	14	52-23	74	88-99	95
18-90	84	99-52	43	2-45	61	74-58	11	41-33	21
47-33	31	21-33	30	80-10	11	58-99	93	49-56	79
45-25	58	40-82	76	89-73	32	44-96	46	74-5	12
96-14	89	2-73	67	54-4	80	39-68	34	52-26	82

24-28	37	66-40	49	40-48	45	17-61	72	73-63	45
70-96	87	68-53	34	59-26	67	46-30	85	17-90	76
45-85	82	95-60	90	96-65	59	94-26	30	69-20	67
67-52	54	12-44	1	85-36	11	42-88	97	76-62	1
91-99	49	80-76	45	55-95	27	66-59	44	81-59	36
23-56	26	13-85	79	48-96	12	73-85	31	22-27	9

pmed4:		**n-100**		**edge-200**		**optimal-3034**			
1-2	45	2-3	23	3-4	96	4-5	59	5-6	24
6-7	78	7-8	24	8-9	70	9-10	5	10-11	66
11-12	93	12-13	31	13-14	67	14-15	89	15-16	93
16-17	22	17-18	83	18-19	83	19-20	89	20-21	59
21-22	11	22-23	85	23-24	24	24-25	94	25-26	79
26-27	3	27-28	53	28-29	62	29-30	36	30-31	77
31-32	14	32-33	65	33-34	13	34-35	72	35-36	44
36-37	84	37-38	17	38-39	11	39-40	98	40-41	70
41-42	80	42-43	61	43-44	35	44-45	5	45-46	51
46-47	35	47-48	4	48-49	8	49-50	43	50-51	19
51-52	63	52-53	46	53-54	97	54-55	73	55-56	23
56-57	87	57-58	37	58-59	15	59-60	28	60-61	9
61-62	78	62-63	27	63-64	92	64-65	81	65-66	82
66-67	30	67-68	67	68-69	88	69-70	54	70-71	76
71-72	16	72-73	60	73-74	95	74-75	10	75-76	86
76-77	76	77-78	9	78-79	82	79-80	94	80-81	6
81-82	72	82-83	84	83-84	41	84-85	80	85-86	31
86-87	29	87-88	24	88-89	76	89-90	84	90-91	35
91-92	71	92-93	28	93-94	57	94-95	76	95-96	50
96-97	25	97-98	92	98-99	44	99-100	46	100-1	10
96-48	8	15-22	31	44-17	76	43-7	56	90-46	49
19-98	62	24-9	68	55-90	80	12-62	70	8-98	38
48-89	58	86-81	76	94-11	93	2-31	46	64-34	54
27-32	33	69-74	68	3-14	62	74-77	27	97-39	57
8-22	56	96-2	33	16-27	24	54-22	53	39-63	57
34-46	12	76-24	48	89-20	57	92-57	80	10-9	17
91-10	86	66-26	96	13-19	60	7-70	38	54-93	98
38-89	18	78-23	64	85-9	3	44-90	31	92-17	55
73-8	9	69-87	18	87-52	33	21-43	20	44-55	31
77-21	74	89-23	77	40-55	7	95-73	96	88-68	87
9-88	59	22-81	24	62-72	70	76-77	19	95-62	70
76-98	14	32-7	67	47-69	75	12-7	91	23-14	63
63-62	42	71-57	60	23-26	62	99-55	10	11-3	71
71-77	99	67-77	84	52-38	81	63-86	45	17-57	76
26-29	36	72-88	37	66-24	48	48-47	15	83-42	99
87-78	77	55-4	57	15-89	31	99-19	24	69-2	97
48-20	63	91-64	33	26-54	61	62-1	89	83-8	67
81-31	71	26-77	36	18-22	28	88-61	73	23-43	70
81-89	91	94-74	54	26-58	22	95-88	83	52-47	35

19-55	48	12-3	68	22-36	67	68-58	18	48-88	58

pmed5:		**n-100**		**edge-200**		**optimal-1355**			
1-2	38	2-3	71	3-4	66	4-5	73	5-6	51
6-7	66	7-8	28	8-9	90	9-10	82	10-11	48
11-12	97	12-13	75	13-14	44	14-15	90	15-16	44
16-17	70	17-18	65	18-19	24	19-20	40	20-21	79
21-22	37	22-23	70	23-24	8	24-25	54	25-26	100
26-27	50	27-28	43	28-29	23	29-30	83	30-31	5
31-32	35	32-33	16	33-34	20	34-35	79	35-36	74
36-37	87	37-38	69	38-39	21	39-40	51	40-41	15
41-42	24	42-43	80	43-44	75	44-45	38	45-46	81
46-47	87	47-48	34	48-49	39	49-50	95	50-51	39
51-52	9	52-53	67	53-54	65	54-55	30	55-56	5
56-57	35	57-58	38	58-59	63	59-60	29	60-61	43
61-62	48	62-63	59	63-64	69	64-65	11	65-66	2
66-67	95	67-68	22	68-69	96	69-70	85	70-71	95
71-72	10	72-73	45	73-74	96	74-75	63	75-76	99
76-77	95	77-78	22	78-79	65	79-80	52	80-81	81
81-82	89	82-83	37	83-84	75	84-85	70	85-86	59
86-87	84	87-88	81	88-89	4	89-90	99	90-91	93
91-92	75	92-93	29	93-94	6	94-95	73	95-96	91
96-97	15	97-98	53	98-99	85	99-100	71	100-1	17
78-97	3	16-69	19	3-99	50	44-5	3	76-28	78
49-50	9	57-52	23	88-24	5	23-86	33	59-15	84
81-64	72	27-36	32	80-49	81	78-59	45	52-66	77
43-33	62	94-3	46	87-19	14	57-10	53	54-99	67
47-73	70	72-61	51	87-46	37	8-29	86	6-68	37
98-41	68	84-91	6	36-3	68	43-10	40	27-81	6
59-8	73	79-74	84	15-96	42	47-75	7	69-66	65
69-15	11	61-41	19	63-72	36	93-26	82	38-63	40
97-22	14	36-7	81	10-51	12	34-87	93	35-75	19
14-12	50	48-81	2	97-52	56	1-60	56	62-94	6
76-74	20	67-27	47	32-45	62	79-16	72	80-49	52
66-40	8	87-83	16	79-49	7	99-93	24	77-14	2
48-14	58	75-44	67	5-96	86	1-69	84	56-69	94
36-11	59	82-36	63	52-96	10	51-90	1	15-60	93
27-29	99	69-72	11	60-27	27	23-15	53	55-10	33
46-97	3	99-48	8	32-60	19	14-44	15	74-88	3
16-97	38	68-69	1	22-92	48	34-51	48	68-25	84
36-54	5	40-57	60	15-7	24	74-78	42	58-57	77
2-94	5	17-41	10	26-36	83	16-48	94	77-83	46
35-68	32	90-59	51	42-45	80	11-19	49	52-79	59

APPENDIX B

EXAMPLE OF RUNS

Example of 5 generations for first replication for the problem pmed1.txt where n=100 number of edges is 200, p=5, popsize=2*n,

Replication.....=1
Generation.....=0
Globalbest.....=6300.00
Optimum.....=5819
Weight.....=0.500
Problem.....=pmed1.txt

Replication.....=1
Generation.....=1
Globalbest.....=6300.00
Optimum.....=5819
Weight.....=0.500
Problem.....= pmed1.txt

Replication.....=1
Generation.....=2
Globalbest.....=6285.00
Optimum.....=5819
Weight.....=0.500
Problem.....= pmed1.txt

Replication.....=1
Generation.....=3
Globalbest.....=6162.00
Optimum.....=5819
Weight.....=0.499
Problem.....= pmed1.txt

Replication.....=1
Generation.....=4
Globalbest.....=6056.00
Optimum.....=5819
Weight.....=0.499
Problem.....= pmed1.txt

Replication.....=1
Generation.....=5
Globalbest.....=6056.00
Optimum.....=5819
Weight.....=0.499
Problem.....= pmed1.txt

Replication.....=1
Generation.....=6
Globalbest.....=5975.00
Optimum.....=5819
Weight.....=0.499
Problem.....= pmed1.txt

Replication.....=1
Generation.....=7
Globalbest.....=5926.00
Optimum.....=5819
Weight.....=0.498
Problem.....= pmed1.txt

Replication.....=1
Generation.....=8
Globalbest.....=5926.00
Optimum.....=5819
Weight.....=0.498
Problem.....= pmed1.txt

Replication.....=1
Generation.....=9
Globalbest.....=5926.00
Optimum.....=5819
Weight.....=0.498
Problem.....= pmed1.txt