A Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem

Mehmet Sevkli[1] and Ali R. Guner[2]

[1] Department of Industrial Engineering, Fatih University, Istanbul, Turkey
msevkli@fatih.edu.tr
[2] Department of Industrial and Manufacturing Engineering
Wayne State University, Detroit, MI, USA
arguner@wayne.edu

Abstract. In this paper, a continuous Particle Swarm Optimization (PSO) algorithm is presented for the Uncapacitated Facility Location (UFL) problem. In order to improve the solution quality a local search is embedded to the PSO algorithm. It is applied to several benchmark suites collected from OR-library. The results are presented and compared to the results of two recent metaheuristic approaches, namely Genetic Algorithm(GA) and Evolutionary Simulated Annealing (ESA). It is concluded that the PSO algorithm is better than the compared methods and generates more robust results.

1 Introduction

The Particle Swarm Optimization (PSO) is one of the recent metaheuristics invented by Eberhart and Kennedy [1] based on the metaphor of social interaction and communication such as bird flocking and fish schooling. In PSO, the potential solutions, so-called particles, move around in a multi-dimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. PSO has been successfully applied to a wide range of applications such as function optimization, neural network training, task assignment, and scheduling problems.

Location problems are one of the most widely studied problems in NP-hard [2] combinatorial optimization problems thus there is a very rich literature in operations research (OR)[3]. In addition, the bank account location problem, network design, vehicle routing, distributed data and communication networks, computer network design, cluster analysis, machine scheduling, economic lot sizing, portfolio management are some instances without facilities to locate problems that is modelled as an UFL problem in the literature.

UFL problems have been studied and examined extensively by various attempts and approaches. All important approaches relevant to UFL problems can be classified into two main categories: exact and metaheuristics based algorithms. There is a variety of exact algorithms to solve the UFL problem, such

M. Dorigo et al. (Eds.): ANTS 2006, LNCS 4150, pp. 316–323, 2006.

as branch and bound [4], linear programming, Lagrangean relaxation [5], dual approach (DUALLOC) of Erlenkotter [6] and the primal-dual approaches of Körkel [7]. Although DUALLOC is an exact algorithm, it can also be used as a heuristic to find good solutions. It is obvious that since the UFL problem is NP-hard [2] exact algorithms may not be able to solve large practical problems efficiently. There are several studies to solve UFL problem with metaheuristics. Some of recent studies are tabu search [8,9], genetic algorithms [10], neighborhood search [11], and simulated annealing [12].

In an UFL problem there are a number of sites, n and a number of customers, m. Each site has a fixed cost fc_i. There is a transport cost from each site to each customer c_{ij}. There is no limit of capacity for any candidate site and the whole demand of each customer has to be assigned to one site. We are asked to find the number of sites to be established and specify those sites such that the total cost will be minimized(1). The mathematical formulation of UFL can be stated as follows [2]:

$$Z = min\left(\sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}.x_{ij} + \sum_{i=1}^{n} fc_i.y_i\right) \quad . \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \quad . \tag{2}$$

$$0 \leq x_{ij} \leq y_i \in \{0;1\} \quad . \tag{3}$$

where

$$i = 1, ..., n; j = 1, ..., m;$$

x_{ij} : the quantity supplied from facility i to customer j;

y_i : whether facility i is established ($y_i = 1$) or not ($y_i = 0$).

Constraint (2) makes sure that all demands have been met by the open sites, and (3) is to keep integrity. Since it is assumed that there is no capacity limit for any facility, the demand size of each customer is ignored, and therefore (2) established without considering demand variable.

The organization of this paper is as follows. Section 2 introduces the proposed PSO algorithm for UFL together with the details about local search procedure. In Section 3 the experimental results are provided and Section 4 presents the conclusions driven.

2 PSO Algorithm for UFL

2.1 A Pure PSO Algorithm

The PSO algorithm proposed here for the UFL problems considers each particle based on three key vectors; position (X_i), velocity (V_i), and open facility (Y_i).

$X_i = [x_{i1}, x_{i2}, x_{i3}, , x_{in}]$ denotes the i^{th} position vector in the swarm, where x_{ik} is the position value of the i^{th} particle with respect to the k^{th} dimension $(k = 1, 2, 3, , n)$. $V_i = [v_{i1}, v_{i2}, v_{i3}, ..., v_{in}]$ denotes the i^{th} velocity vector in the swarm, where v_{ik} is the velocity value of the i^{th} particle with respect to the k^{th} dimension. $Y_i = [y_{i1}, y_{i2}, y_{i3}...y_{in}]$ represents the opening or closing facilities identified based on the position vector(X_i), where y_{ik} represents opening or closing the k^{th} facility of the i^{th} particle. For an *n-facility* problem, each particle contains n number of dimensions.

Initially, the position(x_{ij}) and velocity(v_{ij}) vectors are generated randomly and uniformly as continuous sets of values between (-10.0,+10.0) and (-4.0,+4.0) respectively that is consistent with the literature[13]. The position vector $X_i = [x_{i1}, x_{i2}, x_{i3}, ..., x_{in}]$ corresponds to the continuous position values for n facilities, but it does not represent a candidate solution to calculate a total cost. In order to create a candidate solution, a particle, the position vector is converted to a binary variables, $Y_i \leftarrow X_i$, which is also a key element of a particle. In other words, a continuous set is converted to a discrete set for the purpose of creating a candidate solution, particle. The fitness of the i^{th} particle is calculated by using open facility vector (Y_i). For simplicity, $f_i\,(Y_i \leftarrow X_i)$ is from now on be denoted with f_i.

In order to ascertain how to derive an open facility vector from position vector, an instance of 5-facility problem is illustrated in Table 1. Position values are converted to binary variables using following formula:

$$y_i = \lfloor |x_i mod 2| \rfloor \ . \tag{4}$$

In equation (4) a position value is first divided by 2 and then the absolute value of the remainder is floored; and the obtained integer number is taken as an element of the open facility vector. For example, fifth element of the open facility vector, y_5, can be found as follows: $\lfloor |-5.45 mod 2| \rfloor = \lfloor |-1.45| \rfloor = \lfloor 1.45 \rfloor = 1$.

Table 1. An illustration of deriving open facility vector from position vector for a 5-facility to 6-customer problem

	Particle Dimension(k)				
i_{th} Particle Vectors	1	2	3	4	5
Position Vector(X_i)	1.8	3.01	-0.99	0.72	-5.45
Velocity Vector(V_i)	-0.52	2.06	3.56	2.45	-1.44
Open Facility Vector (Y_i)	1	1	0	0	1

For each particle in the swarm, a personal best, $P_i = [p_{i1}, p_{i2}, p_{i3}, ..., p_{in}]$, is defined, whereby p_{ik} denotes the position value of the i^{th} personal best with respect to the k^{th} dimension. The personal bests are determined just after generating Y_i vectors corresponding to their fitness values. In every generation,t, the personal best of each particle is updated if a better fitness value is obtained. Regarding the objective function, $f_i\,(Y_i \leftarrow X_i)$, the fitness value for the personal best of the i^{th} particle, P_i , is denoted by f_i^{pb}. The personal best vector is initialized with

position vector ($P_i = X_i$) at the beginning. Where $P_i = [p_{i1}, p_{i2}, p_{i3}, ..., p_{in}]$ is the position vector and the fitness values of the personal bests are equal to the fitness of positions,$f_i^{pb} = f_i$.

Then, the best particle with respect to fitness value in the whole swarm is selected with the name global best and denoted as $G_i = [g_1, g_2, g_3, ..., g_n]$. The global best, $f_g = f\,(Y \leftarrow G)$, can be obtained as the best of personal bests over the whole swarm, $f_g = min\{f_i^{pb}\}$, with its corresponding position vector, X_g, which is to be used for $G = X_g$, where G = $[g_1 = x_{g1}, g_2 = x_{g2}, g_3 = x_{g3}, , g_n = x_{gn}]$ and $Y_g = [y_{g1}, y_{g2}, y_{g3}, ..., y_{gn}]$ denotes the Y_i vector of the global best found.

Afterwards, the velocity of each particle is updated based on its personal best and the global best in the following way(5):

$$v_{ik}^{(t+1)} = \left(w.v_{ik}^{(t)} + c_1 r_1 \left(p_{ik}^{(t)} - x_{ik}^{(t)}\right) + c_2 r_2 \left(g_k^{(t)} - x_{ik}^{(t)}\right)\right) \quad . \tag{5}$$

where, w is the inertia weight to control the impact of the previous velocity on the current one. In addition, r_1 and r_2 are random numbers between [0,1] and c_1 and c_2 are the learning factors, which are also called social and cognitive parameters respectively. The next step is to update the positions with (6).

$$x_{ik}^{(t+1)} = x_{ik}^{(t)} + v_{ik}^{(t+1)} \quad . \tag{6}$$

After getting position values updated for all particles, the corresponding open facility vectors are determined with their fitness values in order to start a new iteration if the predetermined stopping criterion is not satisfied. In this study, we apply the *gbest* model of Kennedy and Eberhart [14], which is elaborated in the pseudo code given below.

```
Begin
  Initialize particles positions
  For each particle
    Find open facility vector (4)
    Evaluate(1)
    Do{
      Find the personal best and the global best
      Update velocity(5) and position(6) vectors
      Update open facility vector (4)
      Evaluate(1)
      Apply local search(in (PSO_LS))
    }While (Termination)
End
```

Fig. 1. Pseudo code of *PSO* algorithm for UFL problem

2.2 PSO with Local Search

Apparently, *PSO* conducts such a rough search that it produces premature results, which do not offer satisfactory solutions. For this purpose, it is inevitable

to embed a local search algorithm into PSO so as to produce more satisfactory solutions. In this study, we have employed local search to neighbors of the global best position vector. For the UFL problem, flip operator is employed as a neighborhood structure. Flip operator can be defined as picking one position value of the global best randomly, and then changing its value with using following:

$$g_i = \begin{cases} 0 \leq \rho \leq 1 \ , g_i + 1 \\ 0 \leq \rho \leq 1 \ , g_i - 1 \end{cases}$$

Where ρ is a uniformly generated number between 0 and 1. This function is used for opening a new facility or closing an open one. The local search algorithm applied in this study is sketched in Figure 2. The global best found at the end of each iteration of PSO is adopted as the initial solution by local search algorithm. In order not to loss the best found and to diversify the solution, the global best is randomly modified in which two facilities are flipped based on both random parameters generated, η and κ. Then, flip operator is applied as long as it gets better solution. The final produced one is evaluated and replaced with the old global best if it is better than the initial one.

```
Begin
  Set global best position vector (Y_g) to s_0
  Modify s_0 based on η, κ and set to s
  Set 0 to loop
  Apply Flip to s and get s_1
    if f(s_1) ≤ f(s)
      Replace s with s_1
    else loop=loop+1
  Until (loop < n )
  if f(s) ≤ f(s_0)
    Replace Y_g with s
End.
```

Fig. 2. Pseudo code for local search

3 Experimental Results

This experimental study has been completed in two stages; PSO and PSO_{LS}. Experimental results provided in this section is carried out with two algorithms over 15 benchmark problems that are taken from the OR Library [15], a collection of benchmarks for OR studies. The benchmarks are introduced in Table 2 with their sizes and the optimum values. Although the optimum values are known, it is really hard to hit the optima in every attempt of optimization. Since the main idea is to test the performance of PSO algorithm with UFL benchmark, the results are provided in Table 2 with regard to solution quality indexes: Average Relative Percent Error ($ARPE$) which is as defined in (7), Hit to Optimum Rate (HR) and Computational Processing Time(CPU).

$$ARPE = \sum_{i=1}^{R} \left(\frac{(H_i - U) \times 100}{U_i} \right) / R \tag{7}$$

where H_i denotes the i^{th} replication solution value whereas U is the optimal value provided in literature and R is the number of replications. HR provides

Table 2. Experimental results gained with PSO and PSO_{LS}

			PSO			PSO_{LS}		
Problem	Size	Optimum	$ARPE$	HR	CPU	$ARPE$	HR	CPU
$Cap71$	16×50	932615.75	0.05	0.87	0.12	0.00	1.00	0.01
$Cap72$	16×50	977799.40	0.07	0.80	0.16	0.00	1.00	0.01
$Cap73$	16×50	1010641.45	0.06	0.63	0.27	0.00	1.00	0.01
$Cap74$	16×50	1034976.98	0.07	0.73	0.21	0.00	1.00	0.01
$Cap101$	25×50	796648.44	0.14	0.53	0.67	0.00	1.00	0.08
$Cap102$	25×50	854704.20	0.15	0.40	0.85	0.00	1.00	0.02
$Cap103$	25×50	893782.11	0.16	0.20	1.08	0.00	1.00	0.07
$Cap104$	25×50	928941.75	0.18	0.70	0.47	0.00	1.00	0.02
$Cap131$	50×50	793439.56	0.75	0.07	4.26	0.00	1.00	0.57
$Cap132$	50×50	851495.33	0.78	0.00	4.56	0.00	1.00	0.18
$Cap133$	50×50	893076.71	0.73	0.00	4.58	0.00	1.00	0.42
$Cap134$	50×50	928941.75	0.89	0.10	4.15	0.00	1.00	0.09
$CapA$	100×1000	17156454.48	22.01	0.00	13.64	0.00	1.00	3.03
$CapB$	100×1000	12979071.58	10.75	0.00	16.58	0.00	1.00	5.18
$CapC$	100×1000	11505594.33	9.72	0.00	24.18	0.02	0.50	8.43

the ratio between the number of runs yielded the optimum and the total numbers of experimental trials. The higher HR the better quality of solution, while the lower $ARPE$ the better quality. Obviously, spent CPU for both algorithms are obtained when the best value is got over 1000 iterations for PSO and 250 iterations for PSO_{LS}. All algorithms and other related software were coded in *Borland C++ Builder6* and run on an *Intel Pentium IV 2.6 GHz* PC with *256MB* memory. The parameters used for the PSO algorithm are as follows: The size of the population set to the number of facilities, the social and cognitive parameters were taken as $c1 = c2 = 2$ consistent with the literature[13]. Inertia weight, w, is taken as a random number between 0.5 and 1. For every benchmark suite both algorithms are conducted for 30 replications.

The performance of PSO does not look that impressive as the results produced within the range of time over 1000 iterations. The PSO without local search found 10 optimal solution whereas the PSO with local search algorithm found 15 among 15 benchmark problems. The $ARPE$ index for PSO is very high for *CapA, CapB* and *CapC* benchmarks and none of the attempts hit the optimum value. It may be possible to improve the quality of solutions by carrying on with PSO for further number of iterations, but, then the main idea and useful motivation of employing the heuristics, getting better quality within shorter

time, will be lost. This fact imposed that it is essential to empower PSO_{LS} algorithm to mature the proposed PSO.

The performance of PSO_{LS} algorithm looks very impressive compared to PSO algorithm with respect to all three indexes of solution quality. HR is 1.00 which means 100% of the runs yield with optimum for all benchmark except *CapC*. The experimental study is carried out as a comparative work with a GA

Table 3. Summary of results gained from different algorithms for comparison

	Deviation from Optimum			Average CPU		
Problem	GA	ESA	PSO_{LS}	GA	ESA	PSO_{LS}
$Cap71$	0.00	0.00	0.00	0.287	0.041	0.010
$Cap72$	0.00	0.00	0.00	0.322	0.028	0.010
$Cap73$	0.00033	0.00	0.00	0.773	0.031	0.010
$Cap74$	0.00	0.00	0.00	0.200	0.018	0.010
$Cap101$	0.00020	0.00	0.00	0.801	0.256	0.080
$Cap102$	0.00	0.00	0.00	0.896	0.098	0.020
$Cap103$	0.00015	0.00	0.00	1.371	0.119	0.070
$Cap104$	0.00	0.00	0.00	0.514	0.026	0.020
$Cap131$	0.00065	0.00008	0.00	6.663	2.506	0.570
$Cap132$	0.00	0.00	0.00	5.274	0.446	0.180
$Cap133$	0.00037	0.00002	0.00	7.189	0.443	0.420
$Cap134$	0.00	0.00	0.00	2.573	0.079	0.090
CapA	0.00	0.00	0.00	184.422	17.930	3.030
CapB	0.00172	0.00070	0.00	510.445	91.937	5.180
CapC	0.00131	0.00119	0.00	591.516	131.345	8.430

introduced by Jaramillo et al. [10] and a ESA proposed by Aydin and Fogarty [12].The results of the first two approaches and PSO_{LS} are summarized in Table 3. The performance of PSO_{LS} algorithm looks more impressive compared to the both GA and ESA in both two indexes. Especially, in respect of CPU time PSO_{LS} much more robust than GA. Comparing with ESA, especially for *CapA, CapB* and *CapC* ESA consumes more CPU time than PSO_{LS} algorithm. In conclusion, we can say that PSO_{LS} algorithm is more robust than not only pure PSO algorithm but also both GA and ESA.

4 Conclusion

In this paper, a PSO and a PSO_{LS} algorithm applied to solve UFL problems. The algorithm has been tested on several benchmark problem instances and optimal result are obtained in a reasonable computing time. The results of PSO_{LS} are compared with the results of two recent metaheuristic approaches, namely Genetic Algorithm and Evolutionary Simulated Annealing. It is concluded that the PSO_{LS} algorithm is better than the compared methods and generating more robust results. In addition, to the best of our knowledge, this is the first application of PSO algorithm reported for the UFL in the literature.

References

1. Eberhart, R.C., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In Proc. of the 6^{th} Int. Symposium on Micro Machine and Human Science, Nagoya Japan (1995) 39–43
2. Cornuéjols, G., Nemhauser, G.L.,Wolsey, L.A.: The Uncapacitated Facility Location Problem. Discrete Location Theory Wiley-Interscience, New York (1990) 119-171
3. Mirchandani, P.B., Francis, R.L. (eds.): Discrete Location Theory. Wiley-Interscience, New York (1990)
4. Klose, A.: A Branch and Bound Algorithm for an UFLP with a Side Constraint. Int. Trans. Opl. Res. **5** (1998) 155-168
5. Barcelo, J., Hallefjord, A., Fernandez, E. and Jrnsten, K.: Lagrengean Relaxation and Constraint Generation Procedures for Capacitated Plant Location Problems with Single Sourcing. O R Spektrum **12** (1990) 79-78
6. Erlenkotter, D.: A dual-based procedure for uncapacitated facility location. Operations Research **26** (1978) 992-1009
7. Körkel, M.: On the Exact Solution of Large-Scale Simple Plant Location Problems. European J. of Operational Research,**39(1)**(1989) 157-173
8. K.S. Al-Sultan, Al-Fawzan, M.A.: A tabu search approach to the uncapacitated facility location problem. Annals of Operations Research **86** (1999) 91-103
9. Laurent, M. and Hentenryck, P.V., A Simple Tabu Search for Warehouse Location. European J. of Operational Research **157** (2004) 576-591
10. Jaramillo, J.H., Bhadury, J., Batta, R.: On the Use of Genetic Algorithms to Solve Location Problems. Computers & Operations Research **29** (2002) 761-779.
11. Ghosh, D.: Neighborhood Search Heuristics for the Uncapacitated Facility Location Problem. European J. of Operational Research **150** (2003) 150-162
12. Aydin, M.E., Fogarty, T.C.: A Distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimization Problems. J. of Heuristics **10** (2004) 269-292
13. Shi, Y. Eberhart, R.: Parameter selection in particle swarm optimization. In Evolutionary Programming VIZ: Proc. EP98. Springer-Verlag, New York (1998) 591-600.
14. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm intelligence. Morgan-Kaufmann, San Francisco (2001).
15. Beasley J.E.: OR-Library: http://people.brunel.ac.uk/∼mastjjb/jeb/info.html (2005)