

# A Modified Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem

Sujay Saha<sup>1</sup>, Arnab Kole<sup>2</sup>, and Kashinath Dey<sup>3</sup>

<sup>1</sup> Assistant Professor, CSE Department, Heritage Institute Of Technology, Kolkata, India  
sujay.saha@heritageit.edu

<sup>2</sup> M. Tech Student, CSE Department, Heritage Institute Of Technology, Kolkata, India  
arnab.kole@heritageit.edu

<sup>3</sup> Associate Professor, CSE Department, University Of Calcutta, Kolkata, India  
kndey55@rediffmail.com

**Abstract.** A continuous version of particle swarm optimization (CPSO) is employed to solve uncapacitated facility location (UFL) problem which is one of the most widely studied in combinatorial optimization. The basic algorithm had already been published in the Research Article “A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem” [1]. But in addition to that, the algorithm is slightly modified here to get better result in a lesser time. To make a reasonable comparison, the same benchmark suites that are collected from OR-library [6] are applied here. In conclusion, the results showed that this modified CPSO algorithm is slightly better than the published CPSO algorithm.

**Keywords:** Swarm Intelligence, Continuous Particle Swarm Optimization, UFL, Inertia Factor, Acceleration Coefficient.

## 1 Introduction

Efficient supply chain management has led to increased profit, increased market share, reduced operating cost and improved customer satisfaction for many businesses. One strategic decision is related to physical distribution structure in supply chain management including locating facilities and allocating customers to them is facility location. Facility Location, also known as Location Analysis, is a branch of Operation Research [4] concerning itself with mathematical modeling and solution of problems concerning optimal placement of facilities in order to minimize the transportation costs, avoid placing hazardous materials near housing, outperform competitors’ facilities, etc. We are trying to solve a ‘Uncapacitated Facility Location Problem’ which finds the number of facilities to be established and specify those facilities such that the total cost will be minimized considering the total cost means a fixed cost of setting up a facility in a given site and a transportation cost of satisfying the customer requirements from a facility. We maintain the constraints that there is no limit of capacity for any candidate facility and the whole demand of each customer has to be assigned to one of the facility.

The organization of the paper is as follows: in Section 2, a basic idea is given for Particle Swarm Optimization Technique. Section 3 reports about the definition of the Uncapacitated Facility Location Problem along with the constraints. Section 4 describes the published CPSO algorithm. Section 5 describes the modified CPSO algorithm for solving the Uncapacitated Facility Location problem. Section 6 provides the comparison results of these two algorithms. Finally, Section 7 presents the conclusion driven.

## 2 Particle Swarm Optimization (PSO)

The class of complex systems sometimes referred to as swarm systems is a rich source of novel computational methods that can solve difficult problems efficiently and reliably. When swarms solve problems in nature, their abilities are usually attributed to swarm intelligence [3]; perhaps the best-known examples are colonies of social insects such as termites, bees, and ants. One of the best-developed techniques of this type is particle swarm optimization (PSO). In PSOs, which are inspired by flocks of birds and shoals of fish, a number of simple entities, the particles, are placed in the parameter space of some problem or function, and each evaluates the fitness at its current location. Each particle then determines its movement through the parameter space by combining some aspect of the history of its own fitness values with those of one or more members of the swarm, and then moving through the parameter space with a velocity determined by the locations and processed fitness values of those other members, along with some random perturbations. The members of the swarm that a particle can interact with are called its social neighborhoods. Together the social neighborhoods of all particles form a PSOs social network. More precisely, in the canonical version of PSO, each particle is moved by two elastic forces, one attracting it with random magnitude to the fittest location so far encountered by the particle, and one attracting it with random magnitude to the best location encountered by any of the particle's social neighbors in the swarm. If the problem is  $N$  dimensional, each particle's position and velocity can be represented as a vector with  $N$  components (one for each dimension). Starting with the velocity vector,  $v = (v_1, \dots, v_N)$ , each component,  $v_i$ , is given by:

$$v_i(t+1) = \omega v_i(t) + \psi_1 R_1 (x_{s_i} - x_i(t)) + \psi_2 R_2 (x_{p_i} - x_i(t)) \quad (1)$$

where  $x_{s_i}$  is the  $i$ th component of the best point visited by the neighbors of the particle,  $x_i(t)$  is the  $i$ th component of the particle's current location,  $x_{p_i}$  is the  $i$ th component of its personal best,  $R_1$  and  $R_2$  are two independent random variable uniformly distributed over  $[0, 1]$  is a constant known as the inertia weight, and  $\psi_1$  and  $\psi_2$  are two constants, known as the acceleration coefficients, which control the relative proportion of cognition and social interaction in the swarm. The same formula is used independently for each dimension of the problem, and synchronously for all particles. The position of a particle is updated every time step using the equation:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

The next iteration takes place after all particles have been moved. Eventually, the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to the best location.

### 3 UFL Definition

In a UFL problem, there are a number of customers,  $m$ , to be satisfied by a number of facilities,  $n$ . Each facility has a fixed cost,  $fc_j$ . A transport cost,  $c_{ij}$ , is accrued for serving customer,  $i$ , from facility,  $j$ . There is no limit of capacity for any candidate facility and the whole demand of each customer has to be assigned to one of the facilities. We are asked to find the number of facilities to be established and specify those facilities such that the total cost will be minimized. The mathematical formulation of the problem can be stated as follows:

$$Z = \min( \sum_{i=1}^m \sum_{j=1}^n c_{ij} \cdot x_{ij} + \sum_{j=1}^n fc_j \cdot y_j ). \tag{3}$$

subject to:

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i \text{ in } m. \tag{4}$$

$$0 \leq x_{ij} \leq y_j \quad y_j \in \{0; 1\}. \tag{5}$$

where  $i = 1 \dots m; j = 1, \dots, n; x_{ij}$  represents the quantity supplied from facility  $i$  to customer  $j$ ;  $y_j$  indicates whether facility  $j$  is established ( $y_j = 1$ ) or not ( $y_j = 0$ ). Constraint (4) makes sure that all customers demands have been met by an open facility and (5) 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 (3) established without considering demand variable.

### 4 Earlier Related Work for UFL Problem

According to Sevkli and Guner [1], CPSO considers each particle has three key vectors: position ( $X_i$ ), velocity ( $V_i$ ), and open facility ( $Y_i$ ).  $X_i = [X_{i1}, X_{i2}, \dots, X_{in}]$  denotes the  $i$ th position vector in the swarm, where  $X_{ik}$  is the position of the  $i$ th particle with respect to the  $k$ th dimension. Similarly  $V_i = [V_{i1}, V_{i2}, \dots, V_{in}]$  is the  $i$ th velocity in the swarm, where  $V_{ik}$  is the velocity of the  $i$ th particle with respect to the  $k$ th dimension.  $Y_i$  represents the opening or closing facilities based on the position vector ( $X_i$ ),  $Y_i = [Y_{i1}, Y_{i2}, \dots, Y_{in}]$  where  $Y_{ik}$  represents opening or closing  $k$ th facility of the  $i$ th particle. For an  $n$ -facility problem, each particle contains  $n$  number of dimensions. Initially the positions and velocities are generated as continuous uniform random variables using the following rules:

$$\begin{aligned} x_{ij} &= x_{\min} + (x_{\max} - x_{\min}) \times r_1 \\ v_{ij} &= v_{\min} + (v_{\max} - v_{\min}) \times r_2 \end{aligned} \tag{6}$$

where  $x_{min} = -10.0$ ,  $x_{max} = 10.0$ ,  $v_{min} = -4.0$ ,  $v_{max} = 4.0$ ,  $r_1, r_2 \in [0, 1]$ . The position vectors do not represent a candidate solution to calculate the total cost (fitness value). In order to create a candidate solution the position vector is converted to a binary variable  $Y_i \leftarrow X_i$ , which is also a key element of a particle. This conversion is done using the following formula:

$$y_i = \lfloor \lfloor x_i \rfloor \pmod{2} \rfloor \tag{7}$$

For example, if the position value of one particle with respect to one dimension is -7.47, then the corresponding open facility vector will be

$$\lfloor \lfloor -7.47 \rfloor \pmod{2} \rfloor = \lfloor \lfloor 7.47 \rfloor \pmod{2} \rfloor = \lfloor \lfloor 1.47 \rfloor \rfloor = 1 \tag{8}$$

After generating the position vectors and open facility vectors, the total cost of each particle is calculated as per equation (3) and initially set these cost as local best or personal best (*Pti*) for each particle and the minimum of these personal bests is set as global best (*Gi*) of the swarm. Next, the velocity of each particle is updated as per equation (1) and the position is updated as per equation (2) where  $\omega$  is the inertia factor used to control the impact of the previous velocities on the current one. After updating the position value for all particles, the corresponding open facility vector can be determined to start a new iteration if the predetermined stopping criterion is not yet met. The stopping criteria can be either getting an optimal solution or reaching the maximum number of iterations chosen for obtaining the result in a reasonable CPU time.

## 5 Modifications

To reduce the execution time & the cost, the above algorithm is modified in the following manners: first, to generate the position and the velocity in equation (6), if  $r_1 = 0.5$  then  $x_{ij} = 0$  and similarly if  $r_2 = 0.5$  then  $v_{ij} = 0$ . So instead of doing the addition and multiplication which take some time, the values of both  $x_{ij}$  and  $v_{ij}$  can be set to zero for the above said value of  $r_1$  and  $r_2$ . Secondly, instead of keeping the fixed inertia factor, the value of  $\omega$  can be varied from 0.9 to 0.4 in various iterations to get the lesser optimal cost. So the modified CPSO algorithm is as follows:

### Modified CPSO Algorithm for UFL

Begin

Initialize positions (population) randomly For each particle as:

Generate  $r_1$  randomly in  $[0, 1]$

If  $r_1 = 0.5$  then  $x_{ij} = 0$ , else

$$x_{ij} = x_{min} + (x_{max} - x_{min}) \times r_1$$

End

Initialize velocities randomly For each particle as:

Generate  $r_2$  randomly in  $[0, 1]$

If  $r_2 = 0.5$  then  $v_{ij} = 0$ , else

$$v_{ij} = v_{min} + (v_{max} - v_{min}) \times r_2$$

End

```

Calculate open facility vector as per equation (7)
Calculate fitness value using open facility vector as per equation (3)
Set to position vector and fitness value as personal best (Pti )
Select the best particle and its position vector as global best (Gt)

End
Do {
  For each particle
    Update velocity as per equation (1)
    Update position as per equation (2)
    Find open facility vectors & calculate the fitness value using open facility
    vector
    Update personal best (Pti )
    Update the global best (Gt) value with position vector
  End
  Decrement the value of  $\omega$  by:  $(0.9 - 0.4) / \text{total no. of iterations}$ 
} While (Maximum Iteration is not reached)
    
```

**Table 1.** An illustration of deriving open facility vector from position vector for a 4-customer 3-facility problem

ith particle vectors	Particle dimension (k)		
	1	2	3
Position Vector (Xi)	3.8	-0.93	5.42
Open Facility Vector (Yi)	1	0	1

**Table 2.** An example of 3-facility to 4-customer

Facility Locations		1	2	3
Fixed Cost		15	8	3
Customers	1	12	3	7
	2	2	8	5
	3	4	6	14
	4	9	1	10

Considering the 3-facility to 4-customer problem shown in Table 1, the total cost of open facility vectors can be calculated as follows:

$$\begin{aligned}
 \text{Total Cost} &= \{\text{open facilities fixed cost } (f_{c_j}) + \min(\text{cost of supply from open facilities to customers } i[c_i])\} \\
 &= \{(15+3) + \min(12, 7) + \min(2, 5) + \min(4, 14) + \min(9, 10)\} = \{18+2+7+4+9\} = \{40\}.
 \end{aligned}$$

## 6 Comparison of Results

The modified CPSO algorithm is coded with C language in Linux Platform and run on an Intel Core 2 Duo 2.4 GHz Laptop with 1GB memory. The results are shown in the following table:

**Table 3.** Experimental Results gained with modified CPSO algorithm

Benchmark					
Problems	Size (m x n)	As Per Earlier CPSO Algorithm		As Per Modified CPSO Algorithm	
		Optimal Cost	ACPU Time	Optimal Cost	ACPU Time
Cap 71	16 x 50	932615.75	0.1218	932597.00	0.9700
Cap 72	16 x 50	977799.40	0.1318	977779.00	0.7500
Cap 73	16 x 50	1010641.45	0.1865	1010619.00	0.5900
Cap 74	16 x 50	1034976.98	0.1781	1034956.00	0.5300
Cap 101	25 x 50	796648.44	0.8818	796626.00	1.3800
Cap 102	25 x 50	854704.20	0.7667	854682.00	1.2400
Cap 103	25 x 50	893782.11	0.9938	893758.00	0.9900
Cap 104	25 x 50	928941.75	0.6026	928918.00	0.7700
Cap 131	50 x 50	793439.56	3.6156	794137.00	2.1300
Cap 132	50 x 50	851495.33	3.5599	853149.00	1.8900
Cap 133	50 x 50	893076.71	3.7792	894072.00	1.8900
Cap 134	50 x 50	928941.75	3.3333	928918.00	1.2400

## 7 Conclusion

In this paper, CPSO algorithm is applied to solve UFL problems. We have slightly modified this algorithm for initializing the particle's positions and velocities for a particular value of the random numbers  $r_1$  and  $r_2$  to reduce the average CPU time. Secondly, instead of keeping a fixed  $\omega$  value, we have varied the value in the range of 0.9 to 0.4 over the total number of iterations. This modified algorithm has been tested on 12 files of OR-Library and the better results either in terms of time or in terms of optimal cost are obtained.

## References

1. Guner Ali, R., Mehmet, S.: A Discrete Particle Swarm Optimization Algorithm For Uncapacitated Facility Location Problem. Hindawi Publishing Corporation Journal of Artificial Evolution And Applications 2008, Article ID 861512, 9 (2008), doi:10.1155/2008/861512
2. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
3. Ghosh, D.: Neighborhood search heuristics for the uncapacitated facility location problem. European Journal of Operational Research 150(1), 150–162 (2003)

4. Eberhart, R.C., Kennedy, J.: New optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS 1995), Nagoya, Japan, October 1995, pp. 39–43 (1995)
5. Beasley, J.E.: OR-Library (2005),  
<http://people.brunel.ac.uk/mastjjb/jeb/info.html>