ISSN 2047-3338

PSO and ACO Algorithms Applied to Optimizing Location of Controllers in Wireless Networks

Dac-Nhuong Le
Hanoi University of Science, Vietnam National University, Vietnam
nhuongld@hus.edu.vn

***Abstract*—The optimal location of controllers in wireless networks is an important problem in the process of designing cellular mobile networks. In this paper, we propose two new algorithms based on Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for solving it. Our objective functions are determined by the total distance based on finding maximum flow in a transport network using Ford-Fulkerson algorithm and pheromone matrix of ants satisfies capacity constraints to find good approximate solutions. The experimental results show that our proposed algorithms have achieved much better performance than previous heuristic algorithms.**

***Index Terms*— Terminal Assignment (TA), Optimal Location of Controllers Problem (OLCP), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Wireless Networks**

I. INTRODUCTION

IN the designing of a mobile phone network (*cellular network*) it is very important to place the base stations optimally for a cheaper and better customer service. This issue is related to the problems of location of devices (Base station (BTS), Multiplexers, Switches, etc) [1], [2].

The objective of terminal assignment problem (TA) [3] involves with determining minimum cost links to form a network by connecting a given collection of terminals to a given collection of concentrators. The capacity requirement of each terminal is known and may vary from one terminal to another. The capacity of concentrators is known. The cost of the link from each terminal to each concentrator is also known. The problem is now to identify for each terminal the concentrator to which it should be assigned, under two constraints: Each terminal must be connected to one and only one of the concentrators, and the aggregate capacity requirement of the terminals connected to any concentrator must not exceed the capacity of that concentrator.

The assignment of BTSs to switches (*controllers*) problem introduced in [4]. In which it is considered that both the BTSs and controllers of the network are already positioned, and its objective is to assign each BTSs to a controller, in such a way that a capacity constraint has to be fulfilled. The objective function in this case is then formed by two terms: the sum of the distances from BTSs to the switches must be minimized, and also there is another term related to *handovers*, between cells assigned to different switches which must be minimized. The optimal location of controller problem (OLCP) [5] is selecting N controllers out of M BTSs, in a way that the objective function given by solving the corresponding TA with N concentrators and M-N terminals is minimal.

Both TA and OLCP are Non-Polynomial (NP)-hard optimization problems so heuristic approach is a good choice. In [1], a simulated annealing (SA) algorithm tackled the assignment of cells to controller problem. The results obtained are compared with a lower bound for the problem, and the authors show that their approach is able to obtain solutions very close to the problem's lower bound. Authors in [5] have introduced a hybrid heuristic consisting of SA and a Greedy algorithm for solving the OLCP problem. In [6-7], authors proposed a hybrid heuristic based on mixing genetic algorithm (GA), Tabu Search (TS) to solving the BTS-controller assignment problem in such a way that terminal is allocated to the closest concentrator if there is enough capacity to satisfy the requirement of the particular terminal.

In this paper, we propose two new algorithms based on Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for solving it. Our objective functions are determined by the total distance based on finding maximum flow in a transport network using Ford-Fulkerson algorithm and pheromone matrix of ants satisfies capacity constraints to find good approximate solutions. Numerical results show that our proposed algorithm is much better than previous studies.

The rest of this paper is organized as follows. Section II presents the problem formulation and briefly introduces the main idea of OCLP proposed in [5]. Section III and section IV present our new algorithm for location of controllers in a mobile communication network based on Particle Swarm Optimization and Ant Colony Optimization algorithms. Section V presents our simulation and analysis results, and finally, section VI concludes the paper.

II. PROBLEM FORMULATION

Let us consider a mobile communication network formed by M nodes (BTSs), where a set of N controllers must be positioning in order to manage the network traffic. It is always fulfilled that $N<M$, and in the majority of cases $N \ll M$. We

start from the premise that the existing BTSs infrastructure must be used to locate the switches, since it saves costs. Thus, the OCLP consists of selecting *N* nodes out of the *M* which form the network, in order to locate in them *N* controllers. To define an objective function for the OCLP, we introduce a model for the problem, based on the Terminal Assignment Problem [5].

A. *The Terminal Assignment Problem*

The TA can be defined as follows [3]:

Problem instance:

Terminals: $l_1, l_2, ..., l_{M-N}$

Weights: $w_1, w_2, ..., w_{M-N}$

Concentrators: $r_1, r_2, ..., r_N$

Capacities: $p_1, p_2, ..., p_N$

where, w_i is weight, or capacity requirement of terminal l_i. The weights and capacity are positive integers and

$$w_i < \min \ p_1, p_2, ..., p_N \ , \forall i = 1, 2..., M-N \qquad (1)$$

The terminals and concentrator are placed on the Euclidean grid, i.e., l_i has coordinates (l_{i1}, l_{i2}) and r_j has is located at (r_{j1}, r_{j2}).

Feasible solution: Assign each terminal to one of concentrator such that no concentrator exceeds its capacity. Let $\hat{x} = \hat{x}_1, \hat{x}_2, ..., \hat{x}_{M-N}$ be a vector such that $\hat{x}_i = j$ means that terminal l_i has been assigned to concentrator r_j, with $\hat{x}$ is an integer such that $1 \le \hat{x} \le N$

Capacity of each concentrator must be satisfied:

$$\sum_{i \in R_j} w_i < p_j, \ j = 1..N \qquad (2)$$

where, $R_j = \ i \mid \hat{x}_i = j \ \ $, i.e., R_j represents the terminals that are assigned to concentrator r_j.

Objective function: Find $\hat{x}$ that minimizes:

$$F \ \hat{x} \ = \sum_{i=1}^{M-N} \text{cost} \, t_{ij} \rightarrow \min, \ \ j = 1, 2, ...N \qquad (3)$$

where, $\text{cost} \, t_{ij} = \sqrt{(l_{i1} - r_{j1})^2 + (l_{i2} - r_{j2})^2}$, i.e., the result of the distance between terminal l_i and concentrator r_j. It is important to note that in the standard definition of the TA, there is a major objective (the minimization of the distances between terminals and concentrators), and a major constraint (the capacity constraint of concentrators).

B. *The Optimal Controller Location Problem*

The complete OCLP has to deal with two issues, first, the selection of the *N* controllers in *M* nodes, second for each election, an associated TA. This process can be seen in Figure.1.

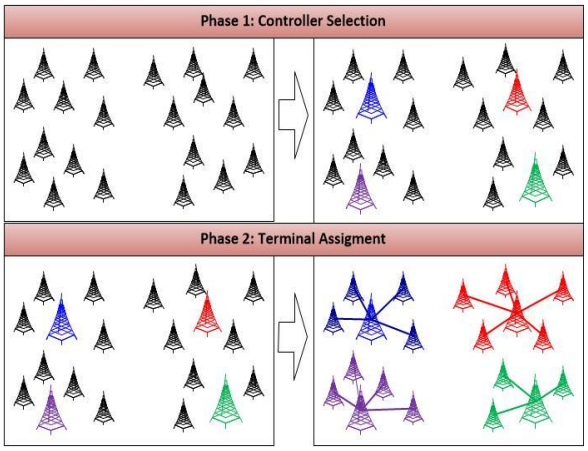


Figure 1. The Optimal Controller Location Problem

Authors in [5] used a Greedy algorithm to obtain this objective function that terminals are consequently allocated to the closest concentrator if there is enough capacity to satisfy the requirement of a particular terminal. If the concentrator cannot handle the terminal, the algorithm searches for the next closest concentrator and performed the same evaluation. The terminals are assigned to concentrators following the order in $\pi \ \ l_{M-N}$ - a random permutation of terminals. That algorithm is called by *SA-Greedy* algorithm.

In [7], the authors considered the following *Lower Bound* (LB) for the TA, as follows:

$$LB = \sum_{i=1}^{M-N} \min_k \ d_{ik} \qquad (4)$$

The Lower Bound comes from the solution obtained by assigning each node *i* to the nearest controller *k*. Hybrid Lower Bound- Greedy algorithm is called by *LB-Greedy* algorithm.

III. PARTICLE SWARM OPTIMIZATION FOR THE OCLP

A. *Particle Swarm Optimization*

Particle swarm optimization (PSO) is a stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy [8-9], inspired by social behavior of bird flocking or fish schooling. It shares many similarities with other evolutionary computation techniques such as genetic algorithms (GA). The algorithm is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO algorithm has no evolution operators such as the crossover and the mutation operator.

In the PSO algorithm, the potential solutions, called particles, fly through the problem space by following the current optimum particle. By observing bird flocking or fish schooling, we found that their searching progress has three important properties. First, each particle tries to move away from its neighbors if they are too close. Second, each particle steers towards the average heading of its neighbors. And the third, each particle tries to go towards the average position of

its neighbors. Kennedy and Eberhart generalized these properties to be an optimization technique as below.

Consider the optimization problem *P*. First, we randomly initiate a set of feasible solutions; each of single solution is a *"bird"* in search space and called *"particle"*. All of particles have *fitness values* which are evaluated by the *fitness function* to be optimized, and have *velocities* which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. The better solutions are found by updating particle's *position*. In iterations, each particle is updated by following two "best" values. The first one is the best solution (*fitness*) it has achieved so far. (The fitness value is also stored.) This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called *gbest*. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called *lbest*

After finding the two best values, the particle updates its velocity and positions with following equation (5) (which use global best gbest) or (6) (which use local best *lbest*) and (7).

$$v[] = v[] + c_1 * rand() * (pbest[] - present[]) + c_2 * rand() * (pbest[] - present[]) \quad (5)$$

$$v[] = v[] + c_1 * rand() * (pbest[] - present[]) + c_2 * rand() * (lbest[] - present[]) \quad (6)$$

$$present[] = present[] + v[] \quad (7)$$

In those above equation, *rand()* is a random number between *0* and *1*; c_1 and c_2 are cognitive parameter and social parameter respectively.

PARTICLE SWARM OPTIMIZATION ALGORITHM

```
{
 FOR each particle
    Initialize particle
 ENDFOR
 DO
    FOR each particle
        Calculate fitness value
        IF the fitness value is better than the
            best fitness value (pBest) in history
            Set current value as the new pBest
        ENDIF
   ENDFOR
   Choose the particle with the best fitness value
 of all the particles as the gBest (or Choose the
 particle with the best fitness value of all the
 neighbors particles as the lBest)
    FOR each particle
      Calculate particle velocity according to(5)or
      (6))
      Update particle position according to (7)
    ENDFOR
 WHILE (STOP CONDITION IS TRUE) }
```

The stop condition mentioned in the above algorithm can be the maximum number of interaction is not reached or the minimum error criteria are not attained.

B. Solving the OCLP based on PSO

In this section, we present application of PSO technique for the OCLP problem. Our novel algorithm is described as follows.

We consider that configurations in the evolution algorithm are sets of *N* nodes which will be evaluated as controllers for the network.

1) Represent and decode a particle: The encoding of the configuration is by means of binary string of length *M*, say $x = x_1, x_2, ...x_M$ where x_i=1 in the binary string means that the corresponding node has been selected to be a controller, whereas a 0 in the binary string means that the corresponding node is not a controller, but serves as BTS. We must select *N* nodes to be the controllers of the network.

2) Initiate population: We use fully random initialization in order to initialize the population. After that, the particle *x* will have *p* 1s.

We present *Particle_Repair* function to ensure that all binary strings in the particles have exactly *N* 1s representing *N* controllers.

PARTICLE REPAIR FUNCTION ALGORITHM

```
Input: The particle x = x1, x2, ...xM  has p 1s
Output: The particle x will have exactly N 1s
IF p<N THEN
     Adds (N-p) 1s in random positions
ELSE
     Select (p-N) 1s randomly and removes them from
     the binary string
```

3) Fitness function: Each particle *x* has exactly *N* 1s representing *N* controllers. We construct a transport network $G = I, J, E$ corresponding particle *x*, where $I = 1, 2, ..., N$ is the set of controllers, $J = 1, 2, ..., M - N$ is the set of BTSs and *E* is the set of edge connections between controller r_i and the BTS l_j.

We find the maximum flow (*max-flow*) of the transport network *G* by adding two vertices *S* (*Source*) and *D* (*Destination*) is shown in Figure 2.

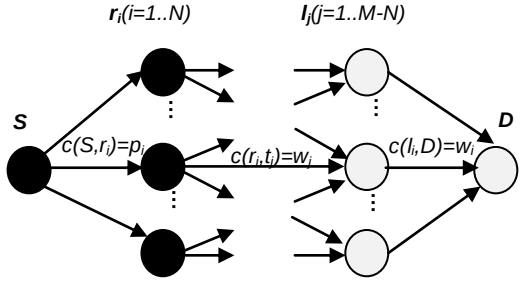


Figure 2. The transport network *G = (I, J, E)* corresponding particle *x*

The weight of the edges on the graph is defined as follows:

- The edges from vertex *S* to the controllers r_i is capacity of r_i, denoted as $c(S,r_i)=p_i$, $(i=1..N)$.
- The edges from BTS l_j to vertex *D* is weight of l_j, denoted as $c(l_j,D)=w_j$, $(j=1..M-N)$.
- The edges from the controllers r_i to the BTSs l_j is denoted as $c(r_i,l_j)=w_i$. ($i, j \in E$).

From the transport network *G*, we find the max-flow satisfies capacity constraints given by the formula (4) based on Ford-Fulkerson algorithm [10].

The fitness value of this particle is computed with the max-flow based on the total distance is given by:

$$F\ x = \sum_{i=1}^{N} \sum_{j=1}^{M-N} \sqrt{(r_{i1} - l_{j1})^2 + (r_{i2} - l_{j2})^2} \qquad (8)$$

4) Stop condition: The stop condition used in this paper is defined as the maximum number of interaction N_{gen} (N_{gen} is also a designated parameter).

IV. Ant Colony Optimization for the OCLP

A. Ant Colony Optimization

The ACO algorithm is originated from ant behavior in the food searching. When an ant travels through paths, from nest food location, it drops pheromone. According to the pheromone concentration the other ants choose appropriate path. The paths with the greatest pheromone concentration are the shortest ways to the food. The optimization algorithm can be developed from such ant behavior.

The first ACO algorithm was the Ant System [11], and after then, other implementations of the algorithm have been developed [12-13].

B. Solving the OCLP based on ACO

In this section, we present application of ACO technique for the OCLP problem. Our new algorithm is described as follows.

We consider that configurations in the evolution algorithm are sets of *N* nodes which will be evaluated as controller for the network. The encoding of the ant *k* configuration is by means of binary string of length *M*, say $k = x_1, x_2, ...x_M$ where x_i=1 in the binary string means that the corresponding node has been selected to be a controller, whereas a 0 in the binary string means that the corresponding node is not a controller, but serve as BTS. We must select *N* nodes to be the controllers of the network.

We use fully random initialization in order to initialize the ant population. After that, the ant *k* will have *p* 1s. We present *Ant_Repair* function to ensure that all binary strings in ants have exactly *N* 1s representing *N* controllers.

In our case the pheromone matrix is generated with matrix elements that represent a location for ant movement, and in the same time it is possible receiver location. Each ant *k* has exactly *N* 1s representing *N* controllers is associated to one matrix.

ANT_REPAIR FUNCTION ALGORITHM

Input: *The ant* $k = x_1, x_2, ...x_M$ *has p 1s*

Output: *The ant k will have exactly N 1s*

```
IF p<N THEN
        Adds (N-p) 1s in random positions
ELSE
        Select (p-N) 1s randomly and removes them
         from the binary string
```

We use real encoding to express an element of matrix A_{m*n} (where *n* is the number of controllers, *m* is number of BTSs).

Each ant can move to any location according to the transition probability defined by:

$$p_{ij}^k = \frac{\left[\tau_{ij}\right]^{\alpha} \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k} \left[\tau_{ij}\right]^{\alpha} \left[\eta_{ij}\right]^{\beta}} \qquad (9)$$

where, τ_{ij} is the pheromone content of the path from controller *i* to BTS *j*, N_i^k is the neighborhood includes only locations that have not been visited by ant *k* when it is at controller *i*, η_{ij} is the desirability of BTS *j*, and it depends of optimization goal so it can be our cost function.

The influence of the pheromone concentration to the probability value is presented by the constant α, while constant β do the same for the desirability. These constants are determined empirically and our values are $\alpha=1$, $\beta=10$.

The ants deposit pheromone on the locations they visited according to the relation.

$$\tau_j^{new} = \tau_j^{current} + \Delta\tau_j^k \qquad (10)$$

where $\Delta\tau_j^k$ is the amount of pheromone that ant *k* exudes to the BTS *j* when it is going from controller *i* to BTS *j*.

This additional amount of pheromone is defined by:

$$\Delta\tau_j^k = \frac{1}{d_{ij}} \qquad (11)$$

In which, d_{ij} is the distance between controller *i* to BTS *j* is given by $d_{ij} = \sqrt{(r_{i1} - l_{j1})^2 + (r_{i2} - l_{j2})^2}$

The cost function for the ant k is the total distance between controllers to BTSs is given by:

$$f_k = \sum_{i=1}^{N} \sum_{j=1}^{M-N} \sqrt{(r_{i1} - l_{j1})^2 + (r_{i2} - l_{j2})^2} \qquad (12)$$

The stop condition we used in this paper is defined as the maximum number of interaction N_{max} (N_{max} is also a designed parameter).

The Figure 3 presents process of our algorithm to solving OCLP based on ACO.

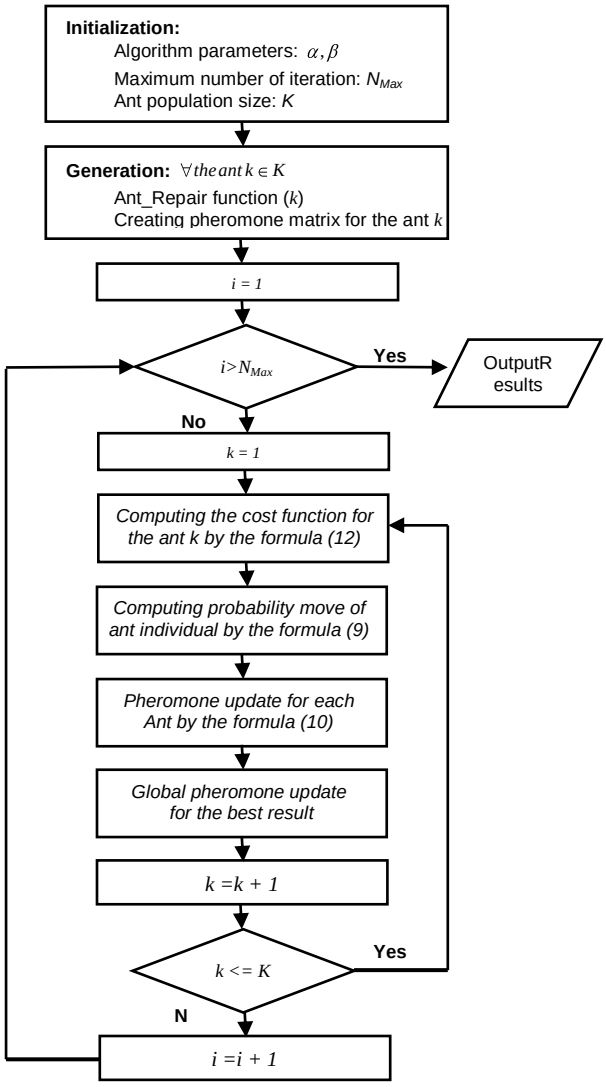


Figure 3. The ant colony algorithm's flow chart

V. Experiments and Results

A. The problems tackled

For the experiments, we have tackled several OCLP instances of different difficulty levels. There are 10 OCLP instances with different values for *N* and *M*, and size networks shown in Table I.

TABLE I. Main characteristic of the problems tackled

Problem #	*Nodes (M)*	*Controllers (N)*	*Grid size*
1	10	2	100x100
2	15	3	100x100
3	20	4	100x100
4	40	6	200x200
5	60	8	200x200
6	80	10	400x400
7	100	15	600x600
8	120	20	800x800
9	150	25	1000x1000
10	200	50	1500x1500

B. PSO algorithm specifications

In our experiments, we have already defined parameters for the PSO algorithm shown in Table II.

TABLE II. The PSO algorithm specifications

Population size	P = 1000
Maximum number of interaction	N_{gen} = 500
Cognitive parameter	c_1 = 1
Social parameter	c_2 = 1
Update population according to	Formula (6) and (7)
Number of neighbor	K = 3

C. ACO algorithm specifications

In our experiments, we have already defined parameters for the ACO algorithm shown in Table III:

TABLE III. The ACO algorithm specifications

Ant Population size	K = 100
Maximum number of interaction	N_{Max} = 500
Parameter	α=1, β=10

D. Numerical Results

Our proposed algorithms (PSO, ACO) are used to optimize location of controllers in wireless networks, and the results are compared with results obtained by SA, SA-Greedy, LB-Greedy.

The experimental results show in Figure 4 (Red color is the best solutions). The objective function of our algorithms has achieved a much better performance than other algorithms. The results show that problems with the small grid size and small number of nodes such as problem #1, #2 and #3, all algorithms has approximate results. However, when the problem size is large, the experimental results are considerable different such as problem #6, #7, #8, #9 and #10.

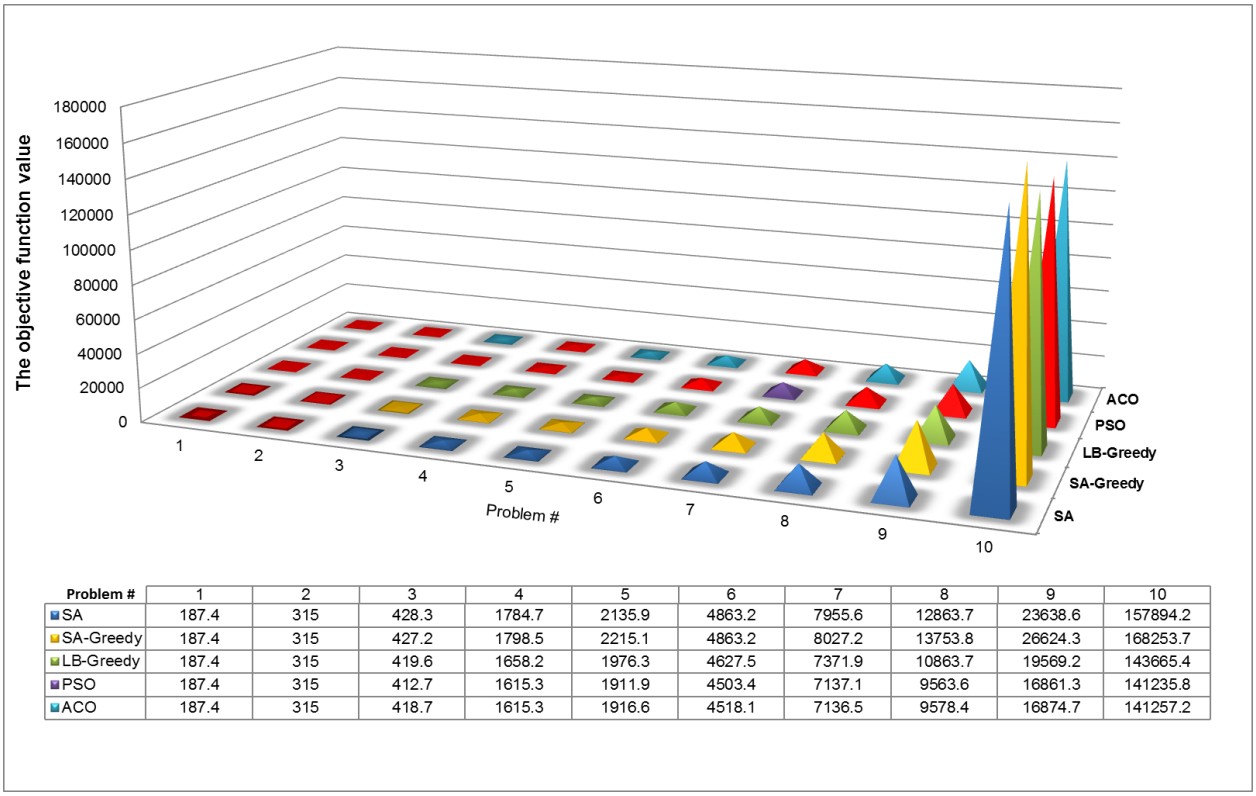
In some cases, *LB-Greedy, SA-Greed, ACO* and *PSO* algorithms choose the same set of nodes to be controllers, but the objective function results of *PSO* or *ACO* are much better. The results show that PSO has better properties compared to ACO algorithm. Figure 5 shows the results of the simulator of solutions for the problem #4 given by SA, SA-Greedy, LB-Greedy, ACO and PSO algorithms.

VI. Conclusion and Future Work

In this paper, we have proposed two new algorithms based on Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for the optimal location of controllers in wireless networks, which is an important problem in the process of designing cellular mobile networks. Our objective functions are determined by the total distance based on finding maximum flow in a transport network using Ford-Fulkerson algorithm and pheromone matrix of ants satisfies capacity constraints to find good approximate solutions. Numerical results show that our proposed algorithm is much better than previous studies.

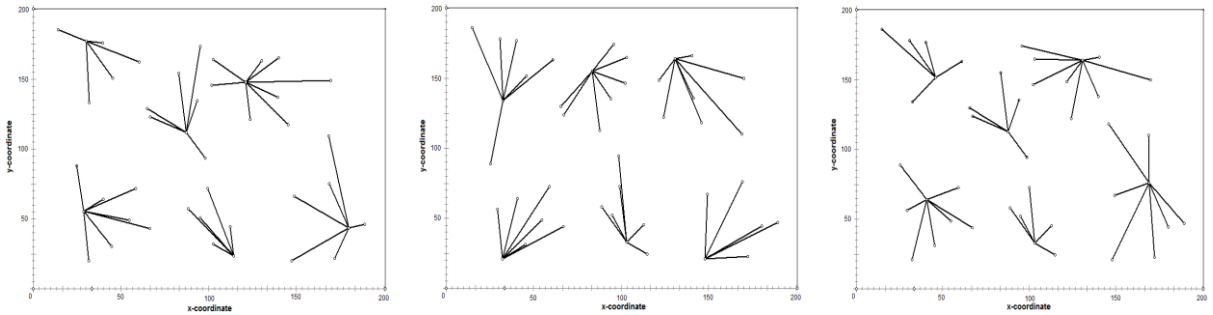
The experimental results show that our proposed algorithm has achieved a much better performance than other heuristic algorithms. It is also proved to be a cost-effective solution. Optimizing location of controllers in wireless networks with profit, coverage area and throughput maximization is our next

research goal.

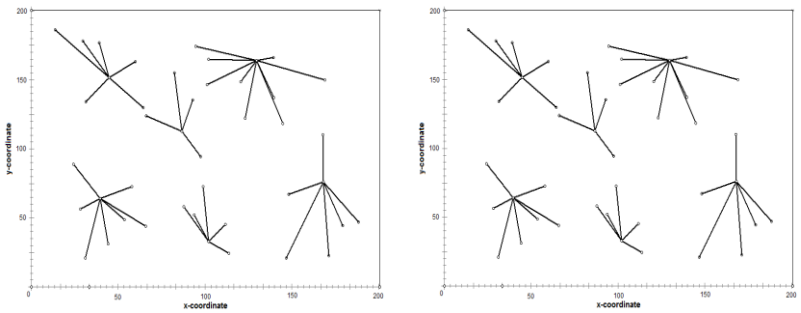


Problem #	1	2	3	4	5	6	7	8	9	10
SA	187.4	315	428.3	1784.7	2135.9	4863.2	7955.6	12863.7	23638.6	157894.2
SA-Greedy	187.4	315	427.2	1798.5	2215.1	4863.2	8027.2	13753.8	26624.3	168253.7
LB-Greedy	187.4	315	419.6	1658.2	1976.3	4627.5	7371.9	10863.7	19569.2	143665.4
PSO	187.4	315	412.7	1615.3	1911.9	4503.4	7137.1	9563.6	16861.3	141235.8
ACO	187.4	315	418.7	1615.3	1916.6	4518.1	7136.5	9578.4	16874.7	141257.2

Figure 4. The results obtained in the OCLP instances tackle



(a) *Simulated Annealing algorithm* (b) *SA-Greedy algorithm* (c) *LB-Greedy algorithm*



(d) *ACO algorithm* (e) *PSO algorithm*

Figure 5. Solutions for the problem #4 given by SA, SA-Greedy, LB-Greedy ACO and PSO algorithms

REFERENCES

[1] B. Krishnamachari and S. Wicker, *Base station location optimization in cellular wireless networks using heuristic search algorithms*, Soft Computing in Communications, L. Wang (Edt), Springer, 2003.

[2] S. Menon and R. Gupta, *Assigning cells to switches in cellular networks by incorporating a pricing mechanism into simulated annealing*, IEEE Trans. Syst. Man Cybern. B, vol. 34, no. 1, pp. 558-565, 2004.

[3] F. N. Abuali, D. A. Schoenefeld and R. L. Wainwright, Terminal assignment in a communications network using genetic algorithms, *In Proc. 22sd Annual ACM Computer Science Conference*, pp. 74-81, ACM press, 1994.

[4] A. Merchant and B. Sengupta, *Assignment of cells to switches in PCS networks*, IEEE/ACM Trans. Networking, vol. 3, no. 5, pp.521-521, 1995.

[5] S. Salcedo-Sanz, A. Portilla-Figueras, S. Jiménez-Fernández, J. A. Martínez-Rojas, *A Hybrid Greedy-Simulated Annealing algorithm for the optimal location of controllers in wireless networks*, Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Databases, Madrid, Spain, pp.159-164, 2006.

[6] F. Houeto and S. Pierre, *A Tabu search approach for assigning cells to switches in cellular mobile networks*, Computer Commun., vol. 25, no. 3, pp. 464-477, 2002.

[7] A. Quintero and S. Pierre, *Assigning cells to switches in cellular mobile networks: a comparative study*, Computers Commun., vol. 26, pp. 950-960, 2003.

[8] James Kennedy and Russell Eberhart, Particle swarm optimization, *in Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, USA, 1995.

[9] J. Kennedy and R. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publisher Inc, 2001.

[10] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey , 1962.

[11] M. Dorigo, V. Maniezzo, and A. Colorni, "*Ant system: Optimization by a colony of cooperating agents*", IEEE Trans. on System, MAN, and Cybernetics-Part B, vol. 26, pp. 29-41, February 1996

[12] E. Rajo-Iglesias, O. Quevedo-Teruel, "*Linear Array Synthesis using an Ant Colony Optimization based Algorithm*", IEEE Trans. on Antennas and Propagation, 2005.

[13] M. Dorigo, M. Birattari, and T. Stitzle, "*Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique*", IEEE computational intelligence magazine, November, 2006

[14] Vinh Trong Le, Nhu Gia Nguyen, Dinh Huu Nghia, "A Novel PSO-Based Algorithm for Gateway Placement in Wireless Mesh Networks", *in Proc,3rd IEEE International Conference on Communication Software and Networks* (ICCSN), China, 2011, pp. 41-46.

[15] Dac-Nhuong Le, Nhu Gia Nguyen, and Vinh Trong Le, *A Novel PSO-Based Algorithm for the Optimal Location of Controllers in Wireless Networks*, International Journal of Computer Science and Network Security (IJCSNS), Vol.12 No.08, pp.23-27, August 30, 2012, Korea.

Dac-Nhuong Le received the BSc degree in computer science and the MSc degree in information technology from College of Technology, Vietnam National University, Vietnam, in 2005 and 2009, respectively. He is a lecturer at the Faculty of information technology in Haiphong University, Vietnam. He is currently a Ph.D student at Hanoi University of Science, Vietnam National University. His research interests include algorithm theory, computer network and networks security.