Performance Evaluation of Heuristic Algorithms for Optimal Location of Controllers in Wireless Networks

Dac-Nhuong Le

Abstract In this paper, we have presented the new Ant Colony Optimization scheme 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 pheromone matrix of ants satisfies capacity constraints to find good approximate solutions. Our pro-posed algorithms may give feasible solutions to this problem based on the global search for high quality feasible solutions. The experimental results show that our pro-posed algorithm has achieved a much better performance than the previous approaches based on heuristic and evolution algorithms.

Keywords Base station controller · Wireless networks · Heuristic · Ant colony optimization

1 Introduction

The size and complexity of computer networks have been growing very fast in the last decade. In the past, a computer network usually only served a small number of devices. Nowadays, it is common to find a computer network that serves hundreds of even thousands of devices. It seems they this incredible growth will continue as the customers needs and the telecommunication technology keep growing. As the size and complexity of computer networks grow, so too does the need for good design strategies. Optimal the placement of base stations in the designing of a wireless network is very important for a cheaper and better customer service. This issue is related to the problems of location of devices [1, 2]. The objective of *Terminal Assignment* (TA) problem [3] involves with determining minimum cost links to form a network by connecting a given collection of terminals to a given

D.-N. Le (✉)
Haiphong University, Haiphong, Vietnam
e-mail: Nhuongld@hus.edu.vn

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*,
Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_84

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 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.

2 Problem Formulation

Let us consider a mobile communication network formed by N nodes (*BTSs*), where a set of M controllers must be positioning in order to manage the network traffic. It is always fulfilled that $M \ll N$, and in the majority of cases. We start from the premise that the existing BTSs infrastructure must be used to locate the switches, since it saves costs. The complete *optimal location of controller problem* (OLCP) has to deal with two issues in Fig. 1. First, the selection of the N controllers in M nodes, second for each selection, an associated TA problem [5].

Let $l_1, l_2, \ldots, l_N$ is set of N *Base Stations* (BTSs), $w_1, w_2, \ldots, w_N$ is the weight requirement of BTS, (l_{j1}, l_{j2}) is the coordinates of BTS $l_j \forall j = 1 \ldots N$ on the Euclidean grid; Let $r_1, r_2, \ldots, r_M$ is set of *Base Station Controllers* (BSCs) should be established, $p_1, p_2, \ldots, p_M$ is the capacity can satisfy of BSC r_i, (r_{i1}, r_{i2}) is the coordinates of BSC $r_i, \forall i = 1 \ldots M$ on the Euclidean grid. The weights and capacity are positive integers and $w_i < \min\{p_1, p_2, \ldots, p_M\}, \forall i = 1, 2 \ldots, N$. Assign each BTS to one of BSC such that no BSC exceeds its capacity. Let $\tilde{x} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_M\}$ be a vector such that means that BTS l_j has been assigned to BSC r_i, with is an integer such that $\tilde{x}_j = i$.

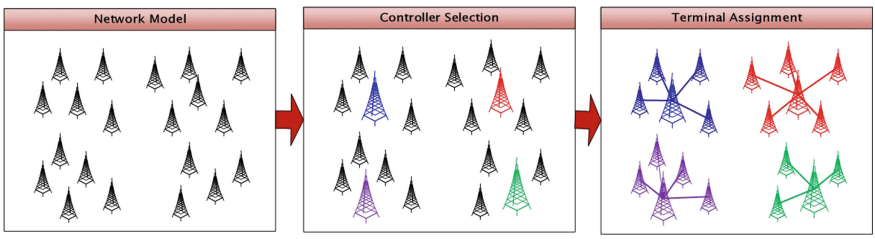

Fig. 1 Optimal location of controller problem model

Capacity of each BSC must be satisfied $\sum_{j \in R_i} w_j < p_i, i = 1 \ldots M$ where, $R_i = \{j | \tilde{x}_j = i\}$, i.e., R_i represents the BTSs that are assigned to BSC r_i. Let $X = \{x_{ij}\}_{M \times (N-M)}$ is a binary matrix describes the connection between the BTS l_j to BSC r_i. Such that, $x_{ij} = 1$ means that BTS l_j has been connected to BSC r_i, and otherwise. The objective of optimal location of controller problem is minimize the total connectivity costs between $(N - M)$ BTSs to M BSCs. The cost of connection from BTS l_j to BSC r_i is calculated by $cost\, t_{ij} = \sqrt{(l_{j1} - r_{i1})^2 + (l_{j2} - r_{i2})^2}$. The problem can be defined as follows:

$$f(\tilde{x}) = \sum_{i=1}^{M} \sum_{j=1}^{N-M} cost\, t_{ij} x_{ij} \rightarrow \min \tag{1}$$

Subject to:

$$\sum_{i=1}^{M} x_{ij} = 1, \quad \forall j = \overline{1 \ldots N - M} \tag{2}$$

$$\sum_{i=j}^{N-M} w_j x_{ij} \leq p_i, \quad \forall i = \overline{1 \ldots M} \tag{3}$$

3 Related Works and Our Works

Both TA and OCLP are *NP-complete* combinatorial optimization problems [1, 6, 7], so heuristic approach is a good choice [8]. All the previous work on the TA provides powerful approaches when the cost of assigning a single terminal to a given concentrator is known before running the algorithms. The cost function is the Euclidean distance between a terminal and its associated concentrator [1]. A *Greedy* is the first algorithm proposed by Abuali et al. in [3] for solving the TA. Khuri and Chui proposed a GA (*Genetic Algorithm*) with a penalty function as an alternative method for solving the TA [4]. They showed its performance by means of the comparison with the greedy algorithm *GA-Greedy* proposed in [3]. The improved GA is proposed include: GENEsYs (*Genetic Search*) [7], LibGA [9], and GGA (*Group Genetic Algorithm*) [10]. In [5], Sanz et al. introduced a hybrid heuristic consisting of SA (*Simulated Annealing*) and a Greedy algorithm for solving the OLCP problem is called by SA-Greedy algorithm. An improvement of *SA-Greedy* is *LB-Greedy* algorithm [11], the lower bound comes from the solution obtained by assigning each BTS l_j to the nearest BSC r_i. A hybrid neural-GA in which a *Hopfield neural network* [12] manages the problems constraints and a GA searches for high quality solutions with the minimum possible cost called by *Hybrid I*, *Hybrid II* proposed in [5, 13].

In the latest works on the OCLP, we proposed GA-BSC [14], *Particle Swarm Optimization* (PSO-BSC) [15] and *Ant Colony Optimization* (ACO-BSC1) [16] algorithms.

4 Our Proposed

In this section, we propose a new ACO algorithm combined *Local search* to improve the speed and quality of solution. 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 [17].

Ant Encoding: We consider that configurations are sets of M nodes which will be evaluated as BSCs for the network. The encoding of the ant k configuration is by means of binary string of length N, say $k = \{x_1, x_2, \ldots x_N\}$ 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 BSC, 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 1 s, we use *Ant_Repair* function to ensure that all binary strings of ants have exactly M 1 s.

Algorithm 1 Ant_repair

Input: The ant $k = \{x_1, x_2, \ldots x_N\}$ has p 1s
Output: The ant k will have exactly M 1s
BEGIN
  IF $p < M$ **THEN** Adds $(M - p)$ 1s in random positions;
  ELSE Select $(p - M)$ 1s randomly and removes them from the binary string;
END.

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 M 1 s representing M BSCs is associated to one matrix. We use real encoding to express an element of matrix $A_{M \times N}$ (where N, M are the number of BTSs, and BSCs). We construct a transport network $G = (I, J, E)$ where $I = \{1, 2, \ldots, M\}$ is the set of BSCs, $J = \{1, 2, \ldots, N - M\}$ is the set of BTSs and E is the set of edge connections between BSC r_i and the BTS l_j. We adding two vertices S (*Source*) and D (*Destination*) is shown in Fig. 2.

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

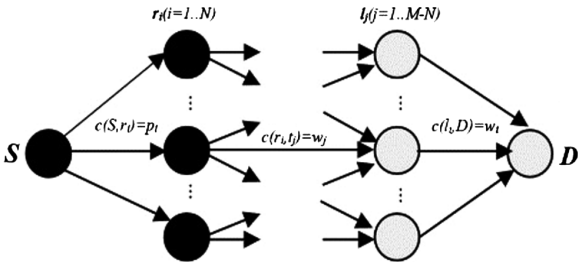

Fig. 2 The transport network $G = (I, J, E)$

$$p_{ij}^{k} = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k}\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}} \tag{4}$$

in which, τ_{ij} is the pheromone content of the path from BSC r_i to BTS l_j, N_i^k is the neighborhood includes only locations that have not been visited by ant k when it is at BSC r_i, η_{ij} is the desirability of BTS l_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.

$$\tau_j^{new} = \tau_j^{current} + \Delta\tau_j^k \quad or \quad \tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{5}$$

where $\Delta\tau_j^k = \frac{1}{\sqrt{(r_{i1}-l_{j1})^2+(r_{i2}-l_{j2})^2}}$ is the amount of pheromone that ant k exudes to the BTS l_j when it is going from BSC r_i to BTS l_j. The cost function for the ant k is the total distance between BSCs to BTSs is given by (1). The stop condition we used in this paper is defined as the maximum number of interaction $N_{\max}$. The pseudo-code of ACO-BSC1 algorithm to solving OCLP as follows:

Algorithm 2 MMAS algortihm to optimizing QoS for multimedia services

```
BEGIN
  Generating the pheromone matrix for the Ant k;
  Update the pheromone values and set x* = k; i =1 ;
  REPEAT
    FOR k = 1 TO K DO
    Computing the cost function for the ant k by the formula (1);
    Computing probability move of ant individual by the formula (4);
    IF f(k) < f(x*) THEN
      Update the pheromone values by the formula (5);
      Set x* = k;
    ENDIF
    ENDFOR
  UNTIL (i > N_Max) or (an acceptable solution is found);
END.
```

ACO-BSC1 algorithm combined with local search is called by ACO-BSC2. The local search algorithm described as follows:

Algorithm 3 Local Search

BEGIN
c_1 = randomly select a BSC in solutions;
c_2 = randomly select a BSC in solutions;
$S = \{candidate_i\}$ by swapping a BTS between BSC c_1 and BSC c_2;
$CurentSolution = candidate_1$;
FOR EACH $candidate_i$ in S **DO**
IF $f(CurentSolution) > f(candidate_i)$ **THEN** $CurentSolution = candidate_i$;
END.

5 Experiments and Results

In our experiments, we have already defined parameters for our algorithms: ant population size $K = 100$, Maximum number of interaction $N_{Max} = 500$, parameter $\alpha = 1$, $\beta = 10$. In order to test the performance of our algorithm, we tackle a set of TA and OCLP instances of different difficulties in 3 case studies. We present the results we have obtained followed by an analysis.

Case study 1: We experiment on 13 test cases in [3]. The coordinates of terminals and concentrators have been randomly obtained in a 100 grid, whereas the weights associated with each terminal were randomly generated $w_j \in [1, 6], \forall j = 1...N$. The capacities of each concentrator assigned fixed $p_i = 12, \forall i = 1...M$. Table 1 shows the comparison of the best objective function of *ACO-BSC1*, *ACO-BSC2*, *Greedy*, *GENEsYs*, *LibGA*, *GGA* algorithms. The results listed under the *Greedy* algorithm are

Table 1 Performance evaluation of the best solution of algorithms in Case study 1

Test	N	M	Greedy	GENEsYs	LibGA	GGA	ACO-BSC1	ACO-BSC2	Improved (%)
#1	100	32	1203	1153	1138	**1115**	**1115**	**1115**	0.88
#2	100	32	1253	1180	**1159**	1166	1166	**1159**	0.94
#3	100	31	1274	1216	1181	**1170**	**1170**	**1170**	1.04
#4	100	33	1438	1394	1344	1359	1344	**1303**	1.35
#5	100	27	1600	1540	1500	1469	1469	**1423**	1.77
#6	100	27	1446	1393	**1373**	1388	1388	**1373**	0.73
#7	100	31	1961	1917	1838	1863	1838	**1725**	2.36
#8	100	27	1865	1803	1702	1781	1630	**1615**	2.50
#9	100	31	1564	1492	1425	1412	1412	**1394**	1.70
#10	100	31	1367	1251	1216	1225	1225	**1182**	1.85
#11	200	93	2002	1939	1898	1919	1769	**1721**	2.81
#12	300	96	2673	2607	2579	2595	2369	**2213**	4.60
#13	400	128	3432	3327	3282	3316	3168	**2775**	6.57

the best solutions yielded by the implementation after 100 executions in each case. We force *Greedy* algorithm to stop after 10,000 iterations in order to make a comparison with the *GENEsYs*, *LibGA*, *GGA* algorithms which also iterate for 10,000 generations. We use the same population size is 500 and the same crossover rate is 0.6 in the three genetic algorithms. The experiment results show that our approach are useful to reach the feasible regions very fast more than the three genetic algorithms. The *GENEsYs*, *LibGA*, *GGA* algorithms may have to wander for a large of generations in the search space before the feasible regions can be identified. The *Greedy* algorithm is the fast algorithm, but it does not always produce near optimal solutions, and the *GENEsYs* does not perform as well as the *LibGA*, *GGA* algorithms in all cases. While, our proposed algorithms run very efficiently and yields feasible solutions consistently based on the heuristic information.

Case study 2: Table 2 summarize the results of executing the *Hybrid I*, *Hybrid II*, *ACO-BSC1* and *ACO-BSC2* algorithms in the best and average cases on 15 test cases after 100 executions in each case. The 15 TA test cases of different sizes, the difficulty increases with the problem size in [5]. The coordinates of terminals and concentrators have been randomly obtained in a 100 grid, whereas the weights associated with each terminal were randomly generated. The capacities of each concentrator vary from one problem to another, being in a range. Experimental results show that the proposed algorithms are found to be optimal solution in the best case similar to the *Hybrid II* algorithm. However, the performance of *ACO-BSC1*, *ACO-BSC2* algorithms equally or better than the *Hybrid I* and *Hybrid II* algorithms in all cases. The discovery of the *ACO-BSC2* algorithm are better demonstrat-ed in the average objective function.

Case study 3: We experiment on 10 OCLP instances of different sizes, the difficulty increases with the problem size. The coordinates of terminals and concentrators have been randomly obtained in Table 3, whereas the weights associated with each terminal were randomly generated $w_j \in [1, 30], \forall j = 1 \ldots N$. The capacities of each concentrator vary from one problem to another, being in a range $p_i \in [50, 150], \forall i = 1 \ldots M$. We compare the results obtained by the objective function of the *SA*, *SA-Greedy*, *LB-Greedy*, *GA-Greedy*, *GA-BSC*, *PSO-BSC*, *ACO-BSC1* and *ACO-BSC2* algorithms in the best and average cases. All experiment are independent of all others, the results listed in Table 4 is the best solutions and average solutions after 100 executions in each case. The experimental results show that the objective function of our algorithms has achieved a much better performance than other algorithms. In the small grid size and small number of nodes such as problem #29, #30 and #31, all algorithms has approximate results both the best solutions and the average solutions. However, when the problem size is large, the experimental results are considerable different such as problem #34, #35, #36, #37 and #38. In some cases, all algorithms choose the same set of nodes to be BSCs, but the objective function results of the *ACO-BSC2* algorithm are much better.

Table 5 shows the computational time of the seven algorithms. The computation time of the *GA-BSC*, *PSO-BSC* is smaller than the *SA*, *SA-Greedy* and *LB-Greedy* algorithm, however it is larger than the *ACO-BSC1* and *ACO-BSC2* algorithms

Table 2 Performance evaluation of the solutions of algorithms in Case study 2

Test	N	M	Σw_j	Σp_i	Hybrid I		Hybrid II		ACO-BSC1		ACO-BSC2	
			$(j = 1...N)$	$(i = 1...M)$	Best	Average	Best	Average	Best	Average	Best	Average
#14	10	3	35	39	**73.8**	**73.8**	**73.8**	**73.8**	**73.8**	**73.8**	**73.8**	**73.8**
#15	10	3	39	42	**86.9**	**86.9**	**86.9**	**86.9**	**86.9**	**86.9**	**86.9**	**86.9**
#16	10	3	34	37	**96.6**	**96.6**	**96.6**	**96.6**	**96.6**	**96.6**	**96.6**	**96.6**
#17	20	6	77	83	**151.1**	152.6	**151.1**	152.8	**151.1**	152.6	**151.1**	**152.3**
#18	20	6	61	68	**164.6**	166.8	**164.6**	168.3	**164.6**	166.8	**164.6**	**165.9**
#19	20	6	72	79	**165.7**	167.1	**165.7**	167.3	**165.7**	167.1	**165.7**	**167.1**
#20	30	10	117	127	295.8	303.4	**295.4**	307.2	**295.4**	303.5	**295.4**	**301.3**
#21	30	10	98	120	**303.1**	318.5	**309.8**	322.1	**303.1**	319.9	**303.1**	**316.9**
#22	30	10	94	120	311.6	321.9	**304.8**	325.9	**304.8**	320.8	**304.8**	**316.1**
#23	50	17	182	204	496.6	517.2	**490.3**	521.8	**490.3**	514.5	**490.3**	**501.8**
#24	50	17	174	193	**516.6**	538.8	521.7	541.5	**516.6**	536.1	**516.6**	**529.9**
#25	50	17	173	204	546.4	571.4	**542.6**	572.2	**542.6**	561.6	**542.6**	**550.2**
#26	100	30	292	360	884.1	925.6	**881.3**	924.4	**881.3**	915.9	**881.3**	**914.4**
#27	100	30	334	360	**815.4**	883.6	823.4	878.9	**815.4**	878.3	**815.4**	**871.8**
#28	100	30	342	360	**849.4**	898.8	862.8	907.4	**849.4**	893.4	**849.4**	**878.3**

Table 3 Main features of OCLP problems tackled

Test	Number of BTSs (N)	Number of BSCs (M)	Grid size	$\Sigma w_j (j = 1...N)$	$\Sigma p_i (i = 1...M)$
#29	10	2	100 × 100	174	195
#30	15	3	100 × 100	298	324
#31	20	4	100 × 100	381	413
#32	40	6	200 × 200	527	638
#33	60	8	200 × 200	962	1150
#34	80	10	400 × 400	1258	1435
#35	100	15	600 × 600	1479	1612
#36	120	20	800 × 800	1581	1835
#37	150	25	1000 × 1000	1793	1908
#38	200	50	1500 × 1500	2384	2571

Table 4 Comparison of the results obtained by the difference algorithms considered

Test	SA		SA-Greedy		LB-Greedy		ACO-BSC1	
	Best	Average	Best	Average	Best	Average	Best	Average
#29	**187.4**	196.3	**187.4**	192.6	**187.4**	189.1	**187.4**	**187.4**
#30	**315.0**	347.6	**315.0**	328.5	**315.0**	335.4	**315.0**	**315.0**
#31	428.3	431.5	427.2	429.8	419.6	428.7	**418.7**	**419.1**
#32	1784.7	1826.3	1798.5	1818.9	1658.2	1735.4	**1615.3**	**1631.5**
#33	2091.3	2135.9	1996.7	2215.1	1954.7	1976.3	**1916.6**	**1945.2**
#34	4625.6	4863.2	4612.4	4863.2	4531.8	4627.5	**4518.1**	**4557.4**
#35	7346.4	7955.6	7536.5	8027.2	7213.7	7371.9	**7136.5**	**7182.9**
#36	12863.7	14769.6	13753.8	14176.8	10863.7	11325.7	**9578.4**	**9621.7**
#37	23638.6	24518.2	26624.3	26875.1	19569.2	18423.6	**16874.7**	**17934.5**
#38	157894.2	167452.1	168253.7	172147.5	143665.4	151763.9	**141257.2**	**14855.8**
Test	GA-BSC		GA-Greedy		PSO-BSC		ACO-BSC2	
	Best	Average	Best	Average	Best	Average	Best	Average
#29	**187.4**	191.4	**187.4**	190.7	**187.4**	188.5	**187.4**	**188.2**
#30	**315.0**	328.3	**315.0**	329.1	**315.0**	327.6	**315.0**	**323.4**
#31	415.4	425.6	417.3	428.5	**412.7**	**416.8**	**412.7**	**416.8**
#32	**1615.3**	1637.2	**1615.3**	1639.5	**1615.3**	1631.9	**1615.3**	**1628.7**
#33	**1910.6**	2027.4	1927.3	2105.8	1911.9	2012.7	**1910.6**	**1934.1**
#34	4507.8	4623.9	4539.3	4681.7	**4503.4**	4572.8	**4503.4**	**4543.9**
#35	7144.1	7352.4	7156.3	7320.2	7137.1	7217.3	**7136.5**	**7161.8**
#36	9584.3	10625.1	9632.5	11150.5	**9563.6**	97121.4	**9563.6**	**9611.2**
#37	16896.7	16912.5	**16861.3**	16894.8	**16861.3**	16878.1	**16861.3**	**16872.2**
#38	141276.9	141362.8	141335.2	141374.1	**141235.8**	141272.8	**141235.8**	**141257.7**

Table 5 Computation time (*in seconds*) of compared difference algorithms

Test	SA	SA-Greedy	LB-Greedy	GA-Greedy	GA-BSC	PSO-BSC	ACO_BSC1	ACO_BSC2
#29	0.4528	0.4278	0.3912	0.3986	0.3979	0.3858	0.3711	**0.3681**
#30	0.5793	0.5284	0.4837	0.4695	0.4788	0.4672	**0.4629**	0.4756
#31	1.1547	1.2719	1.1956	1.1967	1.1859	1.1753	**1.1467**	1.1491
#32	1.9561	1.8260	1.7978	1.8153	1.7547	1.7193	1.6836	**1.6173**
#33	2.3934	2.1868	2.1329	2.1411	2.0972	2.0448	**2.0423**	2.0426
#34	3.0284	2.9525	2.9851	2.8772	2.8561	2.7923	2.7568	**2.6331**
#35	3.6874	3.7356	3.5647	3.5626	3.5723	3.5539	3.5482	**3.5072**
#36	4.2693	4.1388	4.0522	3.9784	3.9841	3.9311	3.9269	**3.6851**
#37	5.1932	5.2191	5.2326	5.2167	5.1871	5.1589	5.1251	**5.0775**
#38	6.5471	6.2833	6.1972	6.1523	6.1365	6.0921	6.0343	**6.0148**

compared. The *ACO-BSC2* algorithm is able to obtain much better solutions than the *ACO-BSC1* algorithm, which is a reasonable computation time. The *ACO-BSC2* is the fastest algorithm, as expected in almost cases.

6 Conclusion and Future Works

In this paper, we have presented the new ACO scheme for the optimal location of controllers in wireless networks. The proposed algorithms overcomes the disadvantages of previous approaches based on Greedy heuristics are no longer valid, and *blind* algorithms are necessary for achieving high quality solutions to the problem. Our algorithms may give feasible solutions to this problem based on the global search for high quality feasible solutions. The experimental results show that our proposed algorithms have achieved a much better performance than previous heuristic algorithms. Optimizing location of controllers in wireless networks with profit, coverage area and throughput maximization is our next research goal.

References

1. Krishnamachari, B., et al.: Base station location optimization in cellular wireless networks using heuristic search algorithms. In: Wang, L. (ed.) Soft Computing in Communications. Springer, Berlin (2003)
2. Menon, S., et al.: Assigning cells to switches in cellular networks by incorporating a pricing mechanism into simulated annealing. IEEE Trans. Syst. Man Cybern. **34**(1), 558–565 (2004)
3. Abuali, F.N., et al.: Terminal assignment in a communications network using genetic algorithms. In: Proceedings of the 22nd Annual ACM Computer Science Conference, pp. 74–81. ACM press (1994)
4. Merchant, A., Sengupta, B.: Assignment of cells to switches in PCS networks. IEEE/ACM Trans. Netw. **3**(5), 521–521 (1995)

5. Sanz. S.S., et al.: A Hybrid Greedy-Simulated Annealing algorithm for the optimal location of controllers in wireless networks. In: Proceedings of the 5th WSEAS Madrid, pp. 159–164 (2006)
6. Glaer, C., Reith, S., Vollmer, H.: The complexity of base station positioning in cellular networks. Discrete Appl. Math. **148**(1), 112 (2005)
7. Back, T.: GENEsYs 1.0 Software distribution and installation notes, Systems Analysis Research Group, LSXI, University of Dortmund, Germany (1992)
8. Gendreau, M., Potvin, J.Y.: Handbook of Meta-heuristics. Springer, Berlin (2010)
9. Corcoran, A.L., Wainwright, R.L.: LibGA: A User-Friendly Workbench for Order-Based Genetic Algorithm Research. In: Proceeding of the 1993 ACM/SIGAPP, pp. 111–117. ACM Press, New York
10. Jong, D., Kenneth, A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Dissertation Abstracts International, University of Michigan (1975)
11. Bernardino, E.M.: A hybrid differential evolution algorithm for solving the terminal assignment problem. Lecture Notes in Computer Science, vol. 5517, p. 179186. Springer, Berlin (2007)
12. Hopfield, J.J., Tank, D.W.: Neural computation of decisions in optimization problems. Biol. Cybern. **52**, 141152 (2007)
13. Sanz, S.S., et al.: Optimal switch location in mobile communication networks using hybrid genetic algorithms. Appl. Soft Comput. **8**(4), 14861497 (2008)
14. Le, D.-N., et al.: A New Evolutionary Approach for the Optimal Location of Controllers in Wireless Networks. In: Proceeding of 2nd ICICM 2012, pp. 81–86. Hongkong, 26–27 Oct 2012
15. Le, D.-N., et al.: A novel PSO-based algorithm for the optimal location of controllers in wireless networks. Int. J. Comput. Sci. Netw. Secur. **12**(8), 23–27 (2012)
16. Le, D.-N.: PSO and ACO algorithms applied to optimizing location of controllers in wireless networks. Int. J. Comput. Sci. Telecommun. **3**(10), 1–7 (2012)
17. Sttzle, T., Ibanez, M.L., Dorigo, M.: A Concise Overview of Application of Ant Colony Optimization. Wiley, Hoboken (2010)