Journal of the Korean Institute of Industrial Engineers
Vol. 33, No. 4, pp. 447-456, December 2007.

An Ant Colony Optimization Approach for the Maximum Independent Set Problem

Hwayong Choi · Namsu Ahn · Sungsoo Park†

Department of Industrial Engineering, KAIST, Daejeon 305-701, Korea

개미 군집 최적화 기법을 활용한 최대 독립 마디 문제에 관한 해법

최화용 · 안남수 · 박성수

한국과학기술원 산업공학과

The ant colony optimization (ACO) is a probabilistic Meta-heuristic algorithm which has been developed in recent years. Originally ACO was used for solving the well-known Traveling Salesperson Problem. More recently, ACO has been used to solve many difficult problems. In this paper, we develop an ant colony optimization method to solve the maximum independent set problem, which is known to be NP-hard. In this paper, we suggest a new method for local information of ACO. Parameters of the ACO algorithm are tuned by evolutionary operations which have been used in forecasting and time series analysis. To show the performance of the ACO algorithm, the set of instances from discrete mathematics and computer science (DIMACS) benchmark graphs are tested, and computational results are compared with a previously developed ACO algorithm and other heuristic algorithms.

Keywords: Maximum Independent Set (MIS), Ant Colony Optimization (ACO), Heuristic Algorithm

1. Introduction

In recent years, Maximum Independent Set Problem (MISP) has attracted much attention because of its applicability on many real world problems. This problem is relevant for many theoretical research areas and practical applications. As an example, we mention the clustering problem for peer to peer mobile wireless networks that can be easily reduced to the problem of finding a maximum independent set of nodes in the network (Gerla and Tsai, 1995). In many cases, exact algorithms designed to solve the problem to optimality cannot guarantee a reasonable computation time. Therefore, to take advantage in a computation time, heuristic approaches can be applied for the MISP.

One of the most powerful heuristic algorithms is the Ant Colony Optimization (ACO). Since the first release of ACO in 1997 (Dorigo and Gambardella, 1997), ACO attracted much interest due to its cooperative learning mechanism and a converging process to a good solution. More recently, ACO has been used to solve many difficult problems such as Quadratic Assignment Problem (Gambardella *et al*., 1999), Data Mining (Parpinelli *et al.*, 2002), Vehicle Routing Problem (Gambardella *et al*., 1999), etc. In these approaches, ACO derived good performances

† Corresponding author : Professor Sungsoo Park, Department of Industrial Engineering, KAIST, Kuseong-dong, Yuseong-gu, Daejeon 305-701, Korea, E-mail : sspark@kaist.ac.kr
Received October 2007; revision received November 2007; accepted November 2007.

compared to other algorithms. However, so far, not many research efforts are found on applying ACO for MISP (Li and Xu, 2003). So, in this paper, improved ACO for MISP will be shown, and the performance will be demonstrated.

This paper is organized as follows. In Section 2, overview of ACO is provided with an explanation on how it was applied to solve the Traveling Salesman Problem (TSP). Section 3 is devoted to explain how the proposed ACO can be used to solve the MISP. In Section 4, to show the performance of the proposed ACO, computational experiments are performed on discrete mathematics and theoretical computer science (DIMACS) benchmarks. Finally, Section 5 contains conclusions and some directions on future research.

2. Background about the Maximum Independent Set Problem

We consider a graph $G=(V, E)$ with vertex set V and edge set E and its complement, $\overline{G}=(V, \overline{E})$ where $\overline{E}=\{(v, w)\in E; v, w\in V, v\neq w\}$. An independent set (or vertex packing or stable set) is a vertex set whose elements are pairwise non-adjacent, i.e., a subset $S\subset V$ is independent if for all $v, w\in S$, the edge $(v, w)\notin E$. A maximum independent set is an independent set of maximum cardinality.

The integer programming formulation of the maximum independent set problem is given as follows:

$$\max \sum_{i=1}^{|V|} c_i x_i,$$

$$\text{s. t.} \quad x_i + x_j \leq 1, \qquad \forall (i, j)\in E,$$

$$x_i\in\{0, 1\}, \qquad i=1, \cdots |V|.$$

Clearly, exact algorithms and heuristic algorithms have both strong points and weak points. Exact algorithms can find the maximum independent set. On the other hand, heuristic algorithms usually find the feasible solutions more quickly than exact algorithms. However the quality of the solutions may not be good.

3. Overview of Ant Colony Optimization

The ACO has been inspired by the observation of real ants' manner of finding a shortest path. The art of finding a shortest path between a food and a nest lies in a cooperative learning mechanism among the ants. For example, if a shortest path for food is blocked by a sudden obstacle, then lots of new paths need to be explored by the ants. When a first group of ants has arrived, paths will be chosen randomly, and the paths will be marked by a certain chemical substance, called pheromone. Since pheromone will be evaporated as time passes, a longer path will be marked with less pheromone and a shorter path will be marked with more pheromone. Since the pheromone lures the ants, when a next group of ants has arrived, a path marked with a higher level of pheromone will be explored more with a high probability. Therefore, a shorter path will be visited further by the ants, and this cooperative behavior of the ants will lead to a new shortest path.

This observation of real ants leads to a new heuristic algorithm, ACO. ACO was first proposed to address the famous traveling salesman problem (TSP). Implementation of the ACO for solving TSP can be addressed briefly as follows. First, M artificial ants are placed on M different cities, which are chosen randomly. Next, to construct a tour, unvisited cities need to be chosen by each ant. This selection is based on a certain probability, whose calculation is a function of the so-called local and global information from the current city. Note that, due to the probabilistic selection of the next unvisited city, premature convergence can be avoided. Let local information, η_{ij} be the reciprocal of distance from the current city i to the next unvisited city j. Use of the reciprocal of distance helps to find an acceptable solution in an early stage. Similarly, let global information, τ_{ij} be the amount of pheromone on edge (i, j) which is connection city i and city j from the precedent ants' tours. At the initial tour, τ_{ij} takes some positive constant, which will be denoted by τ_0. Global information will be used for generating a positive feedback on good solution. Now, for the next unvisited city, say j, the se-

lection possibility of m^{th} ant can be given by

$$P_{ij}^{m}\frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{k\in K}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}},\ \forall i,\ j\in K,\quad m\in M,\ K=V\setminus V_{m} \tag{1}$$

where V_m is the set of already visited cities by m^{th} ant, M is the set of the ants. And $\alpha(\geq 0)$ and $\beta(\geq 0)$ denote the relative importance. This city visiting procedure will be completed when all the cities are visited by each of the M ants. After tours are completed, pheromone on each connecting edge will be updated as

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{m\in M}\frac{Q}{L_m},\ \forall i,j\in V \tag{2}$$

where ρ and Q represent an evaporation ratio $(0\leq\rho\leq 1)$ and a positive constant respectively, and if the edge (*i*, *j*) is used in the tour of m^{th} ant, then L_m is the tour length, otherwise Q/L_m is zero. After pheromones are updated, all M ants die and new M ants will be born again. Based on the updated global information, new tours will be constructed by new M ants. This process will be continued until the number of tours is reached to a certain number, or no ant constructs a shorter tour. The details of ACO for solving TSP can be found in (Dorigo and Gambardella, 1997).

4. Ant Colony Optimization for the Maximum Independent Set Problem

The neighbour set $N_G(v)=\{w\in V\,|\,\{v,w\}\in E\}$ of a vertex $v\in V$ is the set of vertices adjacent to v in G. As usual, the degree in G of vertex *v* is denoted as $d_G(v)$ and we have $d_G(v)=|N_G(v)|$. $N_G(S)$ for $S\subseteq V$ denotes the neighbour set of S, i.e. $N_G(S)=U_{v\in S}N_G(v)\setminus S$.

An independent set in *G* can be constructed as follows. A vertex *v* is chosen arbitrarily in *G*, then *v*, vertices adjacent to *v* and edges incident to *v* are deleted. This graph is denoted as a resulting subgraph G_R of *v* and given in figure1. This process will be continued until no vertex remains in a resulting subgraph G_R. Obviously, the selected vertices forms an independent set of *G*. Finding an independent set using this procedure is used in this paper.

MISP is quite different from TSP, such that there is no concept of path or all nodes need to be included. In MISP, which vertices belong to a maximum independent set is the issue, so, pheromones will be stored on vertices instead of edges.

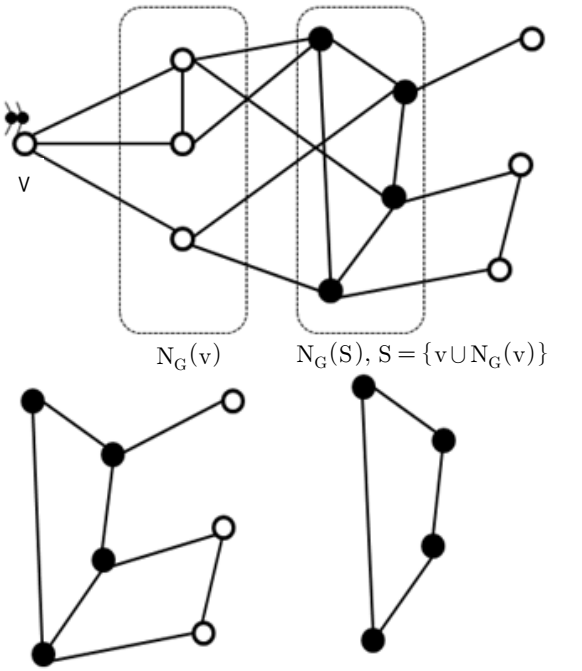

Figure 1. Example of G_R and G_I

4.1 Previous approach of ant colony optimization for the MISP

The ant colony optimization heuristic is used in Li and Xu (2003) in the following way.

In Li and Xu (2003), an approximate approach was used to simulate the ants' manners. Generally, a greedy approach is adopted. For maximum independent set problem, the solution construction procedure of ants is described as follows.

In Li and Xu (2003), they suggest using local information of vertex *v* as

$$\eta_v=\frac{\sum_{i\in N(v)}d_G(i)}{d_G(v)} \tag{3}$$

where $d_G(v)$ is the degree of vertex *v* , and $N_G(v)$ is the neighbour set of vertex *v*. Since the de-

nominator in (Dorigo and Gambardella, 1997) is equal to the number of vertices to be deleted, the numerator in (Dorigo and Gambardella, 1997) is similar to the number of edges to be deleted, main idea is that resulting subgraph of v contains many vertices and a small number of edges. However, this local information has a serious defect. Because of the denominator in (Dorigo and Gambardella, 1997), even though two vertices have the same local information, the resulting subgraphs may have quite different number of vertices and edges.

Although parameters setting are important, Li and Xu (2003) used only one value for each parameter without any explanation about that. Their approach may not fully exploit the advantage of cooperative learning mechanism. That is, even the vertex which belongs to a good solution may have a less chance of growing pheromone.

4.2 New procedure to obtain better local information

By the definition of the independent set, it is clear that if the vertex has a high degree, then it has a less chance of being included in a maximum independent set. However, empirical results indicate that the use of reciprocal of vertex's degree as local information does not generate a good solution. Therefore, a new value, which can be used as local information, is suggested in this paper.

If the vertex v is selected into independent set, $N_G(v)$ can't be selected into the independent set. So vertices in $N_G(S)$, $S=\{v\cup N_G(v)\}$ can be candidates of independent set. We focus on the subgraph G_I induced by $N_G(S)$ about vertex v. In greedy aspect, we know that subgraph G_I induced by $N_G(S)$ which have many vertices and a small number of edges may have many independent vertices. An example of subgraph induced by $N_G(S)$ is shown in <Figure 2>.

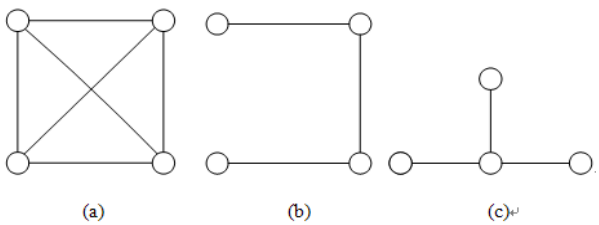

Figure 2. Three example of subgraphs G_I induced by $N_G(S)$

Table 1. Attributes of three subgraphs in figure 2

	Number of vertices	Number of edges	Sum of degree squares	Size of an independent set
(a)	4	6	36	1
(b)	4	3	10	2
(c)	4	3	12	3

Attributes of three subgraphs in figure2 is given in table1. Notice that, to generate a large independent set, it is advantageous if the subgraph induced by $N_G(S)$ have large number of vertices, small number of edges, and large sum of degree squares.

Suppose that, according to the selection of vertices, three different induced subgraphs can be generated as shown in <Figure 2>. Then, (c) should be generated to construct a larger independent set, because (c) has a fewer number of edges than (a) and a larger sum of degree squares than (b).

From this observation, local information of a vertex v is suggested as follows:

$$\eta_v = \frac{(|N_G(S)|+1)\times(\sum_{v\in N_G(S)} d_G^2(v)+1)}{(1/2\sum_{v\in N_G(S)} d_G(v)+1)}, \quad \forall\, v\in V(G_R), \tag{4}$$

where $S=\{v\cup N_G(v)\}$. If $|N_G(S)|$ and $\sum_{v\in N_G(S)} d_G^2(v)$ is zero, local information will be zero. Since $1/2\sum_{v\in N_G(S)} d_G(v)$is the denominator, it must not be zero. Therefore 1 is added on each attributes to guarantee an acceptable local information η_v. This local information is calculated for all vertices in the resulting graph $G_R=G\backslash S$, $S=\{v\cup N_G(v)\}$, where v is the vertex selected in the independent set most recently. And one of vertices which have the largest local information is selected into independent set. Then this procedure is repeated until resulting subgraph G_R is empty.

4.3 Limiting the vertices examined

Empirical results indicate that the local information suggested in this paper is very useful in finding a maximum independent set. However there are many vertices to compute local information and probability. To reduce the computation time, we limit the vertices examined in each

iteration. Therefore, when a new vertex needs to be selected as a vertex in the independent set, we sort the vertices in a decreasing order of their local information, and then we only examine the local information of a vertex when its local information is greater than a certain value. This value is set as follows :

$$\max_{v \in G_R}(\eta_v) \times R,\ 0 \le R \le 1 \tag{5}$$

where G_R is the resulting subgraph, R is the range which is used to indicate a $(R \times 100)$percent of local information.

4.4 Pheromone updates and parameters setting

For the global information, we use the following value:

$$P_v^m = \frac{\tau_v^\alpha \eta_v^\beta}{\sum_{h \in H} \tau_h^\alpha \eta_h^\beta},\ \forall v \in H,\ m \in M,$$
$$H = V \setminus V_m \tag{6}$$

where V_m is the set of already visited vertices by m^{th} ant, M is the set of the ants.

In MISP, pheromone update procedure is similar to the one used for TSP.

$$\tau_v(t+1) = (1-\rho)\tau_v(t) + \sum_{m \in M} T_m \tag{7}$$

where $T_m = \begin{cases} Q \,|S_m|, & \text{if } v \in S_m \\ 0, & otherwise \end{cases}$,

S_m is the generated independent set by m^{th} ant's tour

In order to use ant colony optimization, it is essential to choose parameter values. There are many parameters used in ACO. There are three parameters below which is very important to make a solution.

α : Importance of global information(τ_v)
β : Importance of local information(η_v)
ρ : Evaporation rate of pheromone

These three parameters are in the range of between zero and one. And remaining parameters are the number of ants, Q value, and initial pheromone level($\tau_v(0)$). These don't have much influence in the solution. The number of ants may be larger than the number of nodes of instances. Q value and initial pheromone level have no range, so choosing good value of these parameters is very difficult. Since value of parameters Q and initial pheromone level may dependent on instances, choosing their value may not be meaningful. Therefore we choose only the value of three parameters(α, β, ρ) by evolutionary operation. This evolutionary operation was used in the adaptive control of Exponential smoothing parameters which is the one of the forecasting methods (Montgomery, 1970). Evolutionary operation scheme suggested by Montgomery is based on "simplex" which is the convex hull of k+1 points in general position.

Thus, if there are k parameters, then the number of points in the design is N = k + 1. The N points in R^k correspond to the vertices of a regular-sided simplex. And each vertex of the simplex has values of k parameters in its components. Simplex can be constructed quite easily. Let D be a design matrix, that is, an $N \times k$ matrix whose rows correspond to the vertices of a simplex and whose columns correspond to the k parameters. The design matrix for a simplex design with a starting point can be written as the sum of two $N \times k$ matrices, that is,

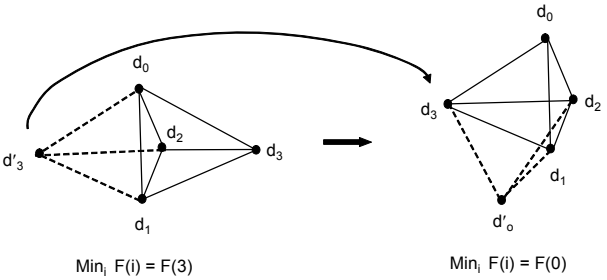

Figure 3. The evolutionary operation suggested by Montgomery.

$$D = \theta \begin{bmatrix} 0 & 0 & 0 \cdots & 0 \\ r & q & q \cdots & q \\ q & r & q \cdots & q \\ \cdots & \cdots & \cdots & \\ q & q & q \cdots & r \end{bmatrix} + \begin{bmatrix} O_1 & O_2 \cdots & O_k \\ O_1 & O_2 \cdots & O_k \\ O_1 & O_2 \cdots & O_k \\ \cdots & \cdots & \cdots \\ O_1 & O_2 \cdots & O_k \end{bmatrix} \tag{8}$$

where $r = \frac{k-1+\sqrt{k+1}}{k\sqrt{2}}$, $q = \frac{\sqrt{k+1}-1}{k\sqrt{2}}$,

θ = the desired edge length of the simplex,
O_i = initial value of i-th parameter, i = 1, 2 ⋯, k

An example of the evolutionary operation with k = 3 is shown in <Figure 3>. Here, F(i) denotes the objective function value of point d_i. In this paper, objective function is the cardinality of in-

dependent set. When the objective value of the vertex j is minimum, the vertex j is deleted and j^* which is an opposite vertex of the vertex j is created. By flipping over, we form a new simplex by deleting vertex j from the design matrix. Here, the new point d_j^* of the simplex can be computed using

$$d_j^* = \frac{2}{k}(d_1 + d_2 + \cdots + d_{j-1} + d_{j+1} + d_{j+2} + \cdots + d_{k+1}) - d_j \quad (9)$$

where d_i is i[th] row of matrix D.

4.5 Procedure of ACO

Procedure of ant colony optimization algorithm we suggested is stated as following

Initialize parameters:
Loop
  For m =1 to M do
  Loop
    Choose a vertex v in G with probability given by (6).
    Set G_R = resulting subgraph of v
  Until G_R is empty
  Save the largest independent set found so far
  Update the pheromone on each vertex given by (7)
Until the termination criteria is satisfied

5. Computational Results

In this section, we report computational experiments performed on instances, which are taken from the Second DIMACS challenge on maximum clique benchmarks. Since finding a maximum clique in a graph is equivalent to searching a maximum independent set in the complement graph, conversions of the benchmarks were performed, and then the experiments are executed.

Parameters(α, β and ρ) which are used in ACO are tuned through the evolutionary operations. D matrix is made by using equation (8) and we use equation obtained from (9) recursively. Four vertices have objective value respectively. And then we can find opposite vertex which has the worst objective value by using equation (9). This operation was repeated until any better solution isn't obtained. We used the value of the desired edge length of the simplex equal to 0.2.

But there is some difficulty. Because this is a cardinality problem of maximum independent set, it is not certain that good solution is obtained from good parameters setting. To overcome this difficulty, we tune parameters through many instances. This procedure is shown in <Table 2>. At first, we found matrix D from on instance, san200_0.9_3 after using evolutionary operation with two iterations. And then the matrix D found was used as initial matrix in the next instance, C125.9. This procedure was terminated when parameters value of matrix D find maximum independent set for a instance, C2000.5.

Table 2. The process of parameters setting

instance	san200_0.9_3			⇒	C125.9			⇒	brock200_4		
	α	β	ρ		α	β	ρ		α	β	ρ
d_0	0.5	0.5	0.5		0.5	0.5	0.5		0.8	0.39	0.53
d_1	0.69	0.69	0.55		0.69	0.55	0.55		0.69	0.55	0.55
d_2	0.55	0.69	0.55		0.65	0.42	0.4		0.65	0.42	0.4
d_3	0.61	0.61	0.37		0.62	0.37	0.59		0.62	0.37	0.59
ineration	after 2 iterations				after 2 interations				after 1 iterations		

⇒	sanr200_0.9			⇒	C250.9			⇒	brock200_2		
	α	β	ρ		α	β	ρ		α	β	ρ
	0.57	0.52	0.71		0.57	0.51	0.52		0.37	0.78	0.67
	0.69	0.55	0.55		0.43	0.38	0.66		0.53	0.69	0.6
	0.49	0.51	0.52		0.49	0.51	0.52		0.49	0.68	0.8
	0.62	0.37	0.59		0.38	0.58	0.67		0.38	0.58	0.67
	after 3 iterations				after 2 interations				after 3 iterations		

⇒	DSJC500.5			⇒	C2000.5			independent set
	α	β	ρ		α	β	ρ	
	0.37	0.78	0.67		0.14	0.67	0.4	15
	0.34	0.67	0.5		0.12	0.69	0.2	15
	0.2	0.66	0.65		**0.27**	**0.79**	**0.3**	**16***
	0.22	0.83	0.54		0.27	0.59	0.3	15
	after 3 iterations				after 4 interations			

Since the other parameters M, Q and $\tau_v(0)$ are not critical for the solution, these parameter are selected from a small set of candidate values by running on a subset of the test instances.

To show the performance of the proposed ACO in this paper, comparisons of computational results with the ACO suggested by Li and Xu (2003) are made. The required parameters for ACO are shown in <Table 3>.

Table 3. The parameters for ACO

	Li and Xu	Choi and Park
M	5	50
α	0.5	0.27
β	0.5	0.79
ρ	0.2	0.3
Q	0.05	1
$\tau_v(0)$	1.0	1.0
R	-	0.1/0.9

The algorithm given in Li and Xu (2003) and ACO in this paper are terminated when it does not produce a better bound in 30, 150 successive iterations respectively. The computational results are shown in <Table 4>. The symbols $|V|$, $|E|$ and $|MIS|$ represent the number of vertices, number of edges, and the best objective value found, respectively. The (*) mark in the $|MIS|$ denotes the cases, where the optimum is reached. In some cases, when the two ACO approaches failed to reach an optimum, better bound is denoted by boldface number.

Graph density is one of the characteristics. Graph density we used is defined as follows :

$$\text{density} = \frac{2|\text{E}|}{|\text{v}|(|\text{v}|-1)} \tag{10}$$

Two approaches are coded in C++ and run on a 2.60-GHz Pentium 4 with 2 GB RAM. In most cases, two ACO approaches find good solutions quickly. However, owing to the limited amount of executable CPU time, if the ACO did not reach to a termination criterion within 1 hour, the program was terminated.

In most cases, optimum or tight bounds can be obtained through the proposed ACO.

The comparisons of computational results in <Table 4> show that proposed ACO in this paper works better than Li and Xu (2003).

To show the performance of ACO in this paper, we compare our computational results with other heuristics that are genetic local search algorithm (GENE), iterated local search algorithm (ITER), multistart local search algorithm (MULT). The results are in <Table 5>. These results are taken from (Marchiori, 2002). Instances are from DIMACS.

We briefly remark the characteristic of the heuristics and import out parameters used in the heuristics in the following :

- Genetic local search algorithm.
  : Simple genetic algorithm + local search
  (Population size = 10, mutation rate = 0.1, crossover rate = 0.9, termination-condition = 2,000 generations)
- Iterated local search algorithm.
  : Repeat local search procedure with just one candidate solution.
  (Population size = 1, mutation rate = 0, termination-condition = 20,000 generations)
- Multistart local search algorithm.
  : Apply local search procedure to each element of large set of candidate solutions.
  (Population size = 20,000, termination-condition = 0 generation)

Three other heuristics are very efficient algorithms to find a maximum independent set. But performance of multistart local search is worse than those of other heuristics and ACO. Genetic local search algorithm and iterated local search algorithm and ACO in this paper showed similar performances. In some cases, ACO finds better solutions than other heuristics. To the contrary, ACO finds poor solutions than other heuristics in some instances.

One interesting fact was observed during the experiments. ACO in this paper use the range R. For dense graphs (i.e. graph density is larger than 0.5), when the range R is 0.9, ACO find good solutions very well. On the contrary, when the range R is 0.1, ACO find good solutions very well in sparse graph (ie. graph density is less than 0.1). This result is shown in <Table 6>. <Table 4>. Comparisons of computational results with Li and Xu (2003).

Table 4. Comparisons of computational results with Li and Xu(2003)

Instance	$\|V\|$	$\|E\|$	Graph density	Optimum	$\|MIS\|$	
					Li and Xu	Choi and Park
brock200_2	200	10024	0.503719	12	9	**11**
brock200_4	200	6811	0.342261	17	15	**16**
brock400_2	400	20014	0.250802	29	**24**	**24**
brock400_4	400	20035	0.251065	33	23	**24**
brock800_2	800	111434	0.348677	24	18	**20**
brock800_4	800	111957	0.350304	26	18	**20**
p_hat300-1	300	33917	0.756232	8	**8***	**8***
p_hat300-2	300	22922	0.511081	25	**25***	**25***
p_hat300-3	300	11460	0.255518	36	35	**36***
p_hat700-1	700	183651	0.750668	11	**11***	**11***
p_hat700-2	700	122922	0.49392	>= 44	**44**	42
p_hat700-3	700	61640	0.246392	>= 62	**60**	57
DSJC500.5	500	62126	0.498004	>= 13	11	**13***
DSJC1000.5	1000	249674	0.499848	>= 15	13	**14**
hamming6-2	64	192	0.095238	32	**32***	**32***
hamming8-4	256	11776	0.360784	16	**16***	**16***
hamming10-4	1024	89600	0.171065	40	33	**37**
keller4	171	5100	0.350877	11	10	**11**
keller5	776	74710	0.248454	27	19	**23**
c-fat200-1	200	18366	0.922915	12	**12***	**12***
c-fat200-2	200	16665	0.837437	24	**24***	**24***
c-fat500-1	500	120291	0.964257	14	**14***	**14***
c-fat500-2	500	115611	0.926741	26	**26***	**26***
san200_0.9_1	200	1990	0.1	70	49	**70***
san200_0.9_2	200	1990	0.1	60	45	**52**
san200_0.9_3	200	1990	0.1	44	32	**35**
san400_0.7_1	400	23940	0.3	40	**22**	**22**
san400_0.7_2	400	23940	0.3	30	**18**	17
san400_0.7_3	400	23940	0.3	22	**14**	**14**
sanr200_0.7	200	6032	0.303116	18	17	**18***
sanr200_0.9	200	2037	0.102362	42	**39**	38
sanr400_0.5	400	39816	0.498947	13	11	**13***
sanr400_0.7	400	23931	0.299887	21	18	**20**
MANN_a27	378	702	0.009852	126	122	**125**
MANN_a45	1035	1980	0.0037	345	338	**342**
C2000.5	2000	999164	0.499832	>= 16	13	**16***

Table 5. Comparisons of computational results with other heuristic algorithms

Instance	Graph density	MULT Avg(best)	GENE Avg(best)	ITER Avg(best)	DIMACS best	ACO-choi Avg(best)
brock200_2	0.503719	12(12)	10.5(12)	10.5(12)	12	11(11)
brock200_4	0.342261	15.7(17)	15.4(16)	15.5(16)	17	15.8(16)
brock400_2	0.250802	21.7(23)	22.5(24)	23.2(25)	25	23.7(24)
brock400_4	0.251065	21.8(22)	23.6(25)	23.1(24)	25	23.7(24)
brock800_2	0.348677	18.0(19)	19.3(20)	19.1(21)	21	19.6(20)
brock800_4	0.350304	18.0(18)	18.9(20)	19.0(20)	21	19.2(20)
p_hat300-1	0.756232	8.0(8)	8.0(8)	8.0(8)	8	8.0(8)
p_hat300-2	0.511081	22.9(25)	25(25)	25(25)	25	25.0(25)
p_hat300-3	0.255518	31.0(32)	34.6(36)	35.1(36)	36	35.1(36)
p_hat700-1	0.750668	9.1(10)	9.8(11)	9.9(11)	11	10.5(11)
p_hat700-2	0.49392	35.5(37)	43.5(44)	43.6(44)	44	41.6(42)
p_hat700-3	0.246392	49.5(52)	60.4(62)	61.8(62)	62	55.4(57)
p_hat1500-1	0.756232	10.2(11)	10.8(11)	10.4(11)	12	10.9(11)
p_hat1500-2	0.511018	46.9(48)	63.8(65)	63.9(65)	65	58.9(60)
p-hat1500-3	0.255518	64.3(67)	92.4(94)	93.0(94)	94	84.5(89)
DSJC500.5	0.498004	12.0(12)	12.2(13)	12.1(13)	15	12.8(13)
DSJC1000.5	0.499848	13.1(14)	13.3(14)	13.5(14)	15	13.7(14)
hamming8-4	0.360784	15.7(16)	16.0(16)	16.0(16)	16	16.0(16)
hamming10-4	0.171065	32.0(33)	37.7(40)	38.8(40)	40	36.1(37)
keller4	0.350877	11.0(11)	11.0(11)	11.0(11)	11	11.0(11)
keller5	0.248454	23.9(25)	26.0(27)	26.3(27)	27	23.7(24)
MANN_a27	0.009852	124.8(125)	125.6(126)	126.0(126)	126	124.5(125)
MANN_a45	0.0037	339.7(340)	342.4(343)	343.1(345)	345	341.3(342)
C125.9	0.101548	32.6(33)	33.8(34)	34.0(34)	34	32.7(33)
C500.9	0.099543	46.7(48)	52.2(56)	52.7(55)	57	50.5(52)
C2000.5	0.499832	14.1(15)	14.2(15)	14.2(15)	16	15.7(16)

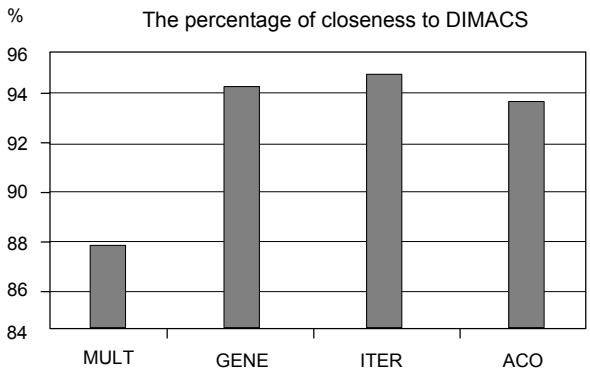

Figure 4. The percentage of closeness to DIMACS.

Table 6. Computational results for different value of R

Instance	graph density	range R		independent set	
MANN_a27	0.0098	0.9	0.1	106	**125**
MANN_a45	0.0037	0.9	0.1	262	**342**
C125.9	0.1015	0.9	0.1	32	**33**
C2000.5	0.4998	0.9	0.1	**16**	14
DSJC500.5	0.4980	0.9	0.1	**13**	12
p_hat700-1	0.7506	0.9	0.1	**11**	10

Local information is calculated in the neighbour set of $S = \{v \cup N_G(v)\}$. For a sparse graph, deviation of local information η_v of vertices is small. Since the number of chosen vertices by local information can be a few, large range R has no meaning. So, for a sparse graph, it is a good approach to give a small value to the range R. On the other hand, for a dense graph, deviation of local information η_v of vertices is large. If the range R is large, vertices having large local information are remaining. Therefore giving a large value to the range R is appropriate to find the maximum independent set.

6. Conclusions and future works

In this research, ACO is modified to solve the MISP, which is a one of the well-known NP-hard problems. Numerical experiments are included to show the performance of the proposed ACO, and the obtained results indicate that the ACO is effective for solving the MISP.

In this research, all the numerical experiments are performed with the parameters found from the proposed evolutionary operation. But this approach isn't complete and optimal. So according to the characteristic of graphs, optimal way of combining the parameter can be a direction for the future research.

At concept of ant colony optimization, ants are scattered at one time. But computer complier scatters one ant at a time. So computation time of proposed ACO in this paper is little slower than other algorithms. In further speed of ACO algorithm can be improved by parallel programming of supercomputer or other methods.

References

Bondy, J. A. and Murty, U. S. R. (1976), *Graph Theory with Application,* American Elsevier Publishing.

DIMACS directories, Second DIMACS implementation challenge.
http://dimacs.rutger.edu/Challenges/index.html

Dorigo, M. and Gambardella, L. M. (1997), *Ant colony system : a cooperative learning approach to the traveling salesman problem*, IEEE Transactions on Evolutionary Computation, **1**(1), 53-66.

Gambardella, L. M., Taillard, E. D., and Dorigo, M. (1999), *Ant Colonies for the Quadratic Assignment Problem*, Journal of the Operational Research Society, **50**(2), 167-176.

Gambardella, L. M., Taillard, E. D., and Giovanni, A. (1999), *A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows*, Advanced Topics In Computer Science Series. Mcgraw-Hill.

Garey, M. and Johnson, D. (1979), *Computers and Intractability- A guide to the Theory of NP-Completeness*, W. H. Freeman and Company.

Gerla, M. and Tsai, J. T-C. (1995), *Multicluster, mobile, multimedia radio network*. Wireless Networks, **1**(3), 255-265.

Li, Y. and Xu, Z. (2003), *An Ant Colony Optimization Heuristic for Solving Maximum Independent Set Problems*, Computational Intelligence and Multimedia Applications, Fifth International Conference 206-211.

Marchiori, E. (2002), *Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem*, Applications of Evolutionary Computing, **2279**.

Montgomery, D. C. (1970), *Adaptive Control of Exponential Smoothing Parameters by Evolutionary Operation*, IIE Transactions, **2**(3), 268-269.

Parpinelli, R., Lopes, S. H. S. and Freitas, A. A. (2002), *Data mining with an ant colony optimization algorithm*, IEEE Transactions on Evolutionary Computation, **6**(4), 321-332.