An Ant Algorithm for the Maximum Clique Problem in a Special Kind of Graph

Received: 10th December 2014; accepted 3rd February 2015

Krzysztof Schiff

DOI: 10.14313/JAMRIS_2-2015/13

Abstract:
The maximum clique problem is a very well-known NP-complete problem of the kind for which meta-heuristic algorithms, which include ant algorithms, have been developed. Well-known instances of problems enable the assessment of the quality of elaborated algorithms; however, there is a particular kind of graph in which each vertex has a nearly equal number of adjacent edges. It is very difficult to find a maximum clique in such a graph. The search for the maximum clique in this particular kind of graph is investigated and compared to the best known ant algorithms.

Keywords*: ant algorithm, maximum clique problem*

1. Introduction

The maximum clique problem was proved to be a NP-complete problem by Karp (1972) [3]; thus there are many meta-heuristic algorithms, to which ant algorithms belong, for this problem. Ant algorithms are quite suitable for combinatorial optimisation problems [1]. The first ant algorithm for the maximum clique problem was presented by Rizzo (2003) [4], the second by Fenet and Solnon (2003) [2], afterwards improved and discussed by Xinshun et al. (2007) [6]. A distributed version of the ant algorithm was formulated by Bui and Rizzo (2004) [7]. This paper shows a new ant algorithm, in which a new dynamic heuristic pattern is used.

2. The Maximum Clique Problem in a special kind of graph

Let G = (V, E) be a graph with a set V of vertices and a set E of edges. A clique C is a subset of set V in which each pair of vertices (v_i, v_j) are linked by an edge e_{ij}. A maximal clique is a clique not included in another clique. The size of clique C is equal to the number of vertices in subset C. The maximum clique is the maximal clique with the greatest number of vertices. Vertex degree d_i is the number of edges adjacent to vertex *i*. A graph with a nearly equal degree for all vertices is a graph in which almost all vertices have the equal vertex degrees $d_{v1} \cong d_{vi} \cong ..... \cong d_{vn}$, v_i V. Such a graph is shown in Fig. 2.1.

The maximum clique problem is an NP-complete problem; hence there are many elaborated meta algorithms for this problem. Vertex degree is the main information used by all meta-heuristic algorithms. In the creation of a maximum clique, a vertex is selected in dependency of all vertex degrees. The order in which these selected vertices are included in the maximum clique C is a very important factor, on which the size of the maximal clique depends. When there are many vertices with equal or nearly equal vertex degrees, there is no useful information for vertex selection and the order in which vertices are included in the maximum clique is not correct. The selection probability formula in ant algorithms includes vertex degrees as useful information, but it is very difficult for ants to make their selection in the correct order, hence they have some difficulty finding the maximum clique in a graph with nearly the same vertex degree for all vertices. Since ants make their vertex selection in dependency of the probability formula they cannot ensure that this is the right order for obtaining the maximum clique. If the probability formula enabled better selection of vertices, then the obtained maximal clique would be closer to the maximum clique. It is therefore very important for the probability formula to enable improved distinction between vertices and improved selection of the one vertex which is the most appropriate in order to ensure the correct selection order for creation of the maximum clique. The probability formula enables ants to better distinguish vertices when the information it gives is more precise.

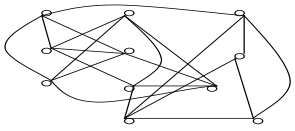

Figure 2.1. A graph with almost equal degree of all vertices

3. The Ant Method

Ants search for the best solution to encountered problems. In order to find this solution, ants communicate among themselves by means of a pheromone *t*. At the beginning of the General Ant Algorithm, which is presented as algorithm 1, a maximal quantity of pheromone is deposited $t(i) = t_{max}$ on all elements *i* M. The set M is the set of elements *i* which can constitute a solution to the given optimisation problem. In the case of the Sudoku problem, set M is the set of all pairs: digit and position. The General Ant Algorithm consists of two main loops, the first connected with the number of cycles, the second with the number of

ants. In each repetition of the first loop, all repetitions of the second loop have to be performed. The best solution S_b found by all ants in one cycle is compared to the best solution S_{best} found by ants in the previous cycle. In each cycle an evaporation mechanism is also used: some of the pheromone evaporates at rate *r* from all elements $i \in M$. In each cycle an additional quantity of pheromone *dt* is also deposited on those elements *i* which constitute solution S_b. When all loops have been done, the best solution is obtained. At the beginning of each inner loop, a starting point is prepared for each ant. From this starting point each ant begins to create a solution to the optimisation problem and then while in the loop each ant selects a next element *j* with probability p(*j*) and adds it to the solution set S. The probability p(*j*) can be expressed by the formula

$$p(j) = \frac{t_j n_j}{\sum_j (t_j n_j)} \quad (3.1)$$

where t_j is the quantity of pheromone deposited on element *j*, $(1 \le j \le \max)$; max is the maximum number of available elements from which the selection can be made; and n_j is a heuristic, that is, the desirability of including element *j* in the solution set S.

Algorithm 1. The General Ant Algorithm

for all i ∈ M: t(i) = t_{max}
for all cycles
  for all ants
    make a starting point
    while (a solution S is not completed) **do**
      check which elements are available to be selected, add them to set A
      select the next element from set A with probability *p*(*j*)
      add a selected element to S
    save in S_b the best solution which has been found by all ants in a cycle
  if S_b is better than S_{best} then save S_b as S_{best} : $S_{best} = S_b$
  for all i: t(i) = t(i) + r* t
  dt = f (S_b)
  if i ∈ S_b then t[i]= t(i) + dt
return S_{best}

This selection can be made only from set A, i.e. from those elements *i* which are available and which can constitute, at this moment of algorithm use, a solution to the optimisation problem. When an element is added to the solution set S, not all elements from set A still satisfy constraints; thus, from the previous set A, a new set A is created by including in this new set A only those elements from the previous set A which satisfy constraints. In the case of the maximal clique problem, when vertex *i* is included in set S, since this vertex *i* is not connected with some vertices from set A, set A should be updated so that all vertices not connected with vertex *i* are removed from set A. Set A should contain only vertices which can be included in solution set S in the future; only these vertices are available for selection.

4. The Proposed Ant Algorithm

The structure of the proposed algorithm is the same as the structure of the ant algorithm developed by Fenet and Solnon, but there is a slight difference between them, since in the proposed ant algorithm there is a desirability function *n* which does not occur in the Fenet and Solnon algorithm.

Algorithm 2.Fenet and Solnon Ant Algorithm.

Choose randomly a first vertex $v_j \in V$
$C \leftarrow v_j$
$Candidates \leftarrow \{v_i \ /(v_i, v_j) \in E\}$
while $Candidates \neq \emptyset$ do
Choose a vertex $v_i \in candidates$ with probability $p(v_i) = \frac{[t_{v_i}]^{\propto}}{\sum_{v_i \varepsilon \text{Candidates}} [t_{v_i}]^{\propto}}$
$C \leftarrow C \cup v_i$
$Candidates \leftarrow Candidates \cap \{v_j /(v_i, v_j) \in E\}$
End while
return S_{best}

The Fenet and Solnon algorithm was improved by Xinshun et al., who proposed an improved formula for selection probability. This pattern is expressed as follows:

$$p(v_i) = \frac{[t_{v_i}]^{\propto}}{\sum_{v_i \varepsilon \text{Candidates}} [t_{v_i}]^{\propto}} + T(time) \frac{d_{v_i}}{EdgesNum} \quad (4.1)$$

where d_{vi} – this is a degree of the v_i vertex

$$d_{v_i} = \sum_{j=1}^{n} d_{ij} \quad (4.2)$$

$d_{ij} = 1$ when $e_{ij} \in E$; else $d_{ij} = 0$. The value of vertex degree d_{vi} is constant since the number of edges e_{ij} adjacent to vertex v_i is constant.

EdgesNum is a number of edges.

$$EdgesNum = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \qquad (4.3)$$

d_{ij}=1 when $e_{ij} \in E$ else d_{ij}=0. T – is a temperature variable, B – is the dumping factor

$$T(time+1) = (1-B)T(time). \qquad (4.4)$$

Their algorithm will be called the Improved Ant Colony Optimisation Algorithm (IACO) later in this paper; at present, in this article, a new formula for selection probability is expressed as follows:

$$p(v_i) = \frac{\left[t_{v_i} n_{v_i}\right]^{\propto}}{\sum_{v_i \varepsilon \text{Candidates}}\left[t_{v_i} n_{v_i}\right]^{\propto}} \qquad (4.5)$$

where $n(v_i)$ – this is a heuristic information about the desirability of vertex v_i,

$$n(v_i) = \frac{D_{v_i}}{m} \qquad (4.6)$$

m – is a number of vertices, D_{vi} – is a vertex degree in Candidates, Candidates – is a set A of available vertices, or rather a graph structure built of vertices from set A and edges which connect vertices from set A,

Degree D_{vi} is computed in a slightly different way than vertex degree d_{vi},

$$D_{v_i} = \sum_{k \varepsilon Candidates} d_{ik} \qquad (4.7)$$

d_{ij}=1 when $e_{ij} \in E$ else d_{ij}=0. D_{vi} is not constant and varies during algorithm operation in accordance with the contents of set A. The factor α is set equal to 3. The pheromone updating method used in the new ant algorithm is the same as in the Fenet and Solnon algorithm. This new proposed algorithm with the new probability formula for vertex selection will henceforth be called the New Ant Colony Optimisation Algorithm (NACO).

5. Results of Experiments

The first experiment concerns average maximum clique size obtained using the New Ant Algorithm and the Improved Ant Algorithm for different values of graph density q {0.962, 0.966, 0.97, 0.974, 0.978, 0.982, 0.986, 0.99, 0.994, 0.998}. Tests were conducted for graphs for a number of vertices equal to n={250, 500}, for a number of cycles equal to lc=200, for a number of ants equal to lm=30 and for an evaporation rate equal to 0.997. Average maximum clique size from 100 measurements and differences in these average maximum clique sizes are shown in Table 5.1 and in Fig. 5.1 for a graph with a number of vertices equal to 500.

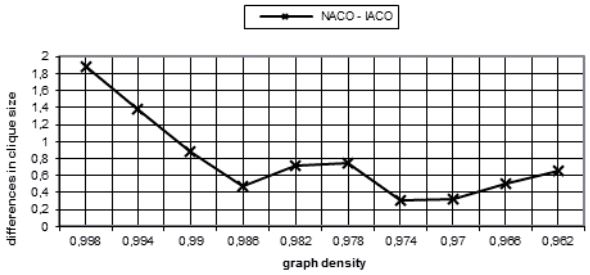

Figure 5.1 Difference in average size for different q and 500 vertices

Table 5.2 Average size, difference in size for different r and n=250, lc=200 lm=30

R	0.999	0.997	0.995	0.993	0.991
IACO	97.28	97.02	97.3	96.85	97.2
NACO	97.82	98.07	98.43	98.96	99.05
NACO – IACO	0.54	1.05	1.13	2.11	1.85

The following experiments were conducted for a constant number of vertices equal to n=250, for a constant graph density equal to q=0.978, and for all other parameters, which varied, including an evaporation rate r, for a number of ants lm and for a number of cycles lc. The results of these experiments: average maximum clique sizes obtained through 100 measurements and differences in these average maximum clique sizes, obtained by using the IACO and NACO algorithms, are shown in Tables 5.2, 5.3, and 5.4 and in Figs. 5.2, 5.3, and 5.4.

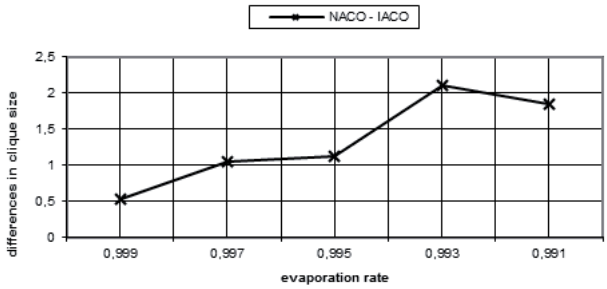

Figure 5.2 Difference in size for different r and for n=250, lc=200 lm=30

Table 5.1 Average size, difference in size for different q and 500 vertices

Q	0.998	0.994	0.99	0.986	0.982	0.978	0.974	0.97
IACO	362	250.8	198.08	165.55	142.78	126.85	114.4	104.74
NACO	363.88	252.18	198.97	166.02	143.5	127.6	114.71	105.06
NACO-IACO	1.88	1.38	0.89	0.47	0.72	0,75	0.31	0.32

Table 5.3 Average size, difference in size for different lm, n=250, lc=200 r=0.997

Lm	10	20	30	40	50
IACO	96.97	97.22	97.3	97.42	97.86
NACO	97.65	98.12	98.43	98.37	98.5
NACO – IACO	0.68	0.9	1.13	0.95	0.64

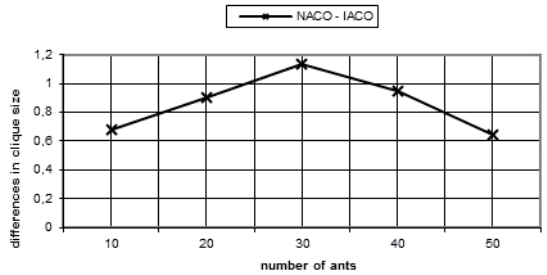

Figure 5.3 Difference in size for different lm and for n=250, lc=200, r=0.997

Table 5.4 Average size, difference in size for different lc, n=250, lm=30, r=0.997

Lc	100	200	300	400	500
IACO	94.35	97.3	97.91	99.19	99.15
NACO	96.16	98.43	99.07	100.67	100.66
NACO – IACO	1.81	1.13	1.16	1.48	1.51

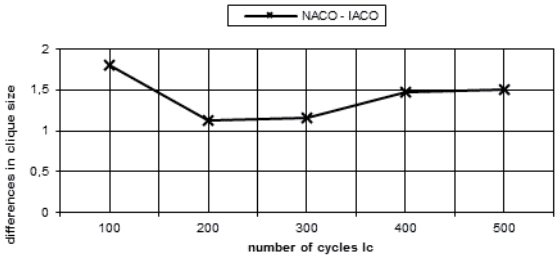

Figure 5.4 Difference in size for different lc and for n=250, lm=30, r=0.997

6. Conclusion

Experiments have shown that the NACO algorithm exhibits permanent superiority over the IACO algorithm for graphs with nearly equal degrees of vertices as regards the size of the maximum clique. In the NACO algorithm the selection probability formula possesses a classical form with a desirability pattern, without a cooling formula as in the IACO algorithm.

AUTHOR

Krzysztof Schiff – Department of Automatic Control and Information Technology, Faculty of Electrical and Computer Engineering, Cracow University of Technology, ul. Warszawska 24, 31-155 Kraków, Poland.

E-mail: kschiff@pk.edu.pl.

REFERENCES

[1] Dorigo M. et al., "Ant algorithms for discrete optimization", *Artificial Life*, vol. 5, no. 2, 1999, 137–172. DOI: 10.1162/106454699568728.

[2] Fenet S., Solnon C., "Searching for maximum cliques with ant colony optimization", *Applications of Evolutionary Computing*, LNCS 2611, Springer, 2003, 236–245. DOI: 10.1007/3-540-36605-9_22.

[3] Karp R.M., "Reducibility among Combinatorial Problems". In: Miller R.E. and Thatcher J.W. (ed.), *Complexity of Computer Computation*, Plenum Press, N.Y., 1972, 85–103. DOI: 10.1007/978-1-4684-2001-2.

[4] Rizzo J.R., *An Ant System Algorithm for Maximum Clique*, Master's Thesis, 2003, The Pennsylvania State University The Graduate School Capital College.

[5] Thang N. et al., "Finding Maximum Cliques with Distributed Ants". In: Deb K. et al. (eds.): *GECCO 2004, LNCS 3102*, 2004, 24–35.

[6] Xinshun Xu et al., An Improved Ant Colony Optimization for the Maximum Clique Problem. In: Third International Conference on Natural Computation (ICNC 2007), vol. 4, 24–27 Aug. 2007, Haikou, 766–770. DOI: 10.1109/ICNC.2007.205.

[7] Bui T.N., Rizzo J.R., "Finding Maximum Cliques with Distributed Ants". In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, Seattle, June 2004, Lecture Notes in Computer Science, vol. 3102, Springer-Verlag Heidelberg, pp. 24–35. DOI: 10.1007/978-3-540-24854-5_3.