Control and Cybernetics

vol. **50** (2021) No. 2

An ant algorithm for the maximum number of 3-cliques in 3-partite graphs*

by

Krzysztof Schiff

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

Abstract: The problem of finding the maximum number of d-vertices cliques ($d = 3$) in d-partite graph ($d = 3$) when graph density q is lower than 1 is an important problem in combinatorial optimization and it is one of many NP-complete problems. For this problem a meta-heuristic algorithm has been developed, namely an ant colony optimization algorithm. In this paper a new development of this ant algorithm and experimental results are presented. The problem of finding the maximum number of 3-vertices cliques can be encountered in computer image analysis, computer vision applications, automation and robotic vision systems. The optimal solution of this problem boils down to finding a set of 3-vertices cliques in a 3-partite graph and this set should have cardinality as high as possible. The elaborated ant colony algorithm can be easily modified for d-dimensional problems, that is for finding the maximum number of d-vertices cliques in a d-partite graph.

Keywords: ant colony optimization, three-partite graph, 3-clique, combinatorial optimization, graph theory

1. Introduction

A *graph* $H = (V, E)$ consists of a finite set V of vertices and a finite set E of edges. A graph is *d-partite* if the set V can be divided into d disjoint subsets V_i, so that $V = \sum_{\iota} V_i$, and there are no edges between the vertices from the same subset V_i. A 3-partite graph is a graph $H_3 = (V_1, V_2, V_3, E)$, having the property mentioned before. A triple (v_1,v_2,v_3) is a 3-vertices *clique* if any two vertices from this set are connected by an edge. In our case, we consider, in a simplified notation, $v_1 \in V_1$, $v_2 \in V_2$ and $v_3 \in V_3$, each pair of these vertices

*Submitted: December 2020; Accepted: February 2021; This paper is an extended version of the paper, presented at the conference BOS/SOR2020 in December 2020.

being connected by an edge e_{ij}, where $i \neq j$ and i, j $\in$\{1,2,3\}. We denote by n the number of vertices in V_i. As announced in the paper title, we would like to find the maximum number of 3-vertices cliques in a 3-partite graph with n vertices in eachV_i, i.e. all of these cliques, in fact. Each of the sets V_i could, of course, have a different number of vertices n_i.

When the graph H is *complete*, this means that there are edges between all pairs of vertices (in the case of d-partite graph: from different subsets V_i). In such a case, there is no problem with finding the maximum number of, say, triple cliques, as these cliques can be easily found. The problem starts to be very hard to solve when the graph H is not complete, but has some density q<1, which means that not every edge exists in such a graph. Any edge is generated with probability q.

This is one of many NP complete problems, see, for instance, Karp (1972) and Knuth (1997). In such a graph, a certain number of triple cliques can be found very easily, but we cannot be sure whether we cannot find yet more triple cliques, so if we look for the maximum cardinality set of triple cliques we should use some optimization algorithm, and the ant colony algorithm is a perfect choice for such a combinatorial optimization problem, see Dorigo (1999). There are no exact polynomial time algorithms for this problem, but there are many heuristic algorithms, elaborated for this purpose, see, e.g., Biro and McDemid (2010), Eriksson (2006) and Chen (2012). Yet, the ant colony heuristics have shown their superiority over the other ones for this problem, see Schiff (2018). The presentation of application of the ant colony optimization algorithm for the problem is provided in Schiff (2018, 2020). The present paper proposes a heuristics, which is used in another part of the ant algorithm than the one that was described in Schiff (2020).

2. The maximum number of 3-vertices cliques in the 3-partite graph

Let H = (V_1, V_2, V_3, E) be a 3-partite graph, that is, the one, where V_1, V_2 and V_3 have no vertex inside of them. Thus, an edge $e_{ij} = (v_i,\, v_j)$ connects two vertices from different sets of vertices, $v_i \in V_i$and $v_j \in V_j$. Graph density, $q \leq$1, is the quotient of the number of edges that actually exist and the maximum number of edges, i.e. when all the pairs of vertices (here: from different subsets V_i) are connected, and it is interpreted as the probability of encountering an edge (between the subsets) in the graph.

We know that in a complete 3-partite graph with n vertices in each part, that is – in a graph with the density q = 1.0, we can easily find n 3-vertices cliques, this means: the maximum number of 3-vertices cliques. A 3-partite graph with such 3-vertices cliques is shown in Fig. 1. When 3-partite graph has the density lower than 1, especially, when a 3-partite graph has the density close to 0, it is no longer possible to find n 3-vertices cliques in it. We can easily find some number of 3-vertices cliques, but usually this is not the maximum number of the cliques, which exist in such a 3-partite graph having density q.

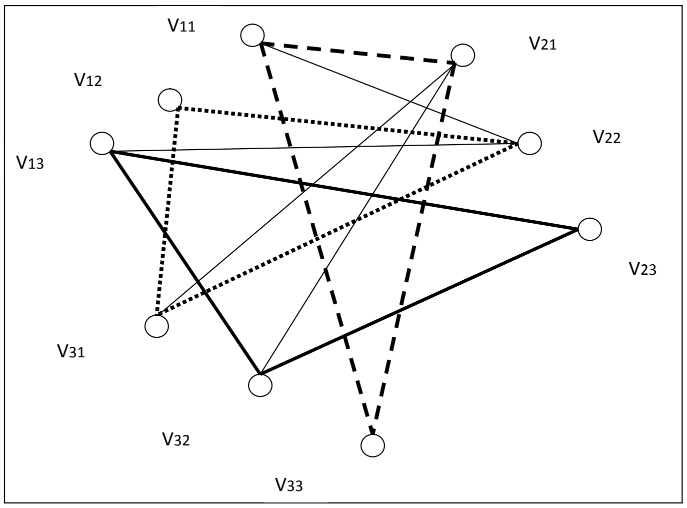

Figure 1. An example of a 3-partite graph with 3-vertices cliques

3. The ant method

The Ant Colony Optimization method is used for finding the best solution of various problems by – roughly – mimicking the search-and-find behavior of the ant colonies. In order to find such a solution ants communicate between themselves by means of a pheromone, whose quantity is denoted t. At the beginning of the General Ant Algorithm, which is presented here as Algorithm 1, on all elements that can form the solution to the problem solved, $i \in M$, the maximum quantity of pheromone is deposited, $t(i) = t_{max}$. The set M is the set of elements i, which can constitute a solution to the considered optimization problem.

Each ant selects an element i from the set M and adds it to the set S, which is empty at the start of algorithm execution. The set S is the current set of elements i, which constitute together the (current version of the) solution to the problem. Each element i from the set M is in some way related to other elements i from the set M. When an ant selects an element i from the set M, some other elements i from the set M cannot be included in the set S. For example, when an ant constructs a clique, it selects a vertex and thereafter it can select the subsequent vertex only from the neighborhood of the already selected vertices; hence, at the beginning, with all vertices being in the set M, an ant makes a start point by selecting a vertex, so only these vertices from the set M, which are in its neighborhood, can be put into the set A.

At the beginning, the set A is empty. This set contains only these elements, which can be potentially selected and inserted into the set S. When a start point is selected, only these elements from the set M are put into the set A, which can be considered when an ant selects the next element during the construction of solution. When some next element is selected by an ant and added to the set S, only these vertices from the set A, which are in the neighborhood of these vertices from the set S can be put into the new set A and the other ones should be removed from the set A. Each ant selects the next element with probability $p(j)$ from among all the elements, which are in the set A and adds it to the set S. The set S_b is one of the sets S, which have been created by ants during one cycle, constituting the best solution to the problem in this cycle. The set S_{best} is the best one of all the sets S_b, which have been created during all the already executed cycles.

In the case of the problem of finding the maximum number of 3-vertices cliques in a 3-partite graph, the set M is the set of all pairs $(v_i,\ v_j)$, where vertices v_i and v_j belong to different parts of the 3-partite graph. The General Ant Algorithm consists of two embedded loops: the first loop concerns the counting of cycles and the second loop concerns the counting of ants. The best solution S_b, which has been found by the ants in one cycle, is compared with the best solution S_{best}, which had been found by the ants in the previous cycles.

In each cycle also an evaporation mechanism is used: some of the pheromone is evaporated with the rate r from all elements $i \in M$. In each cycle, as well, an additional quantity of the pheromone, dt, is deposited on these elements i, which constitute the solution S_b, according to the following formula:

$$dt = 1/[1 + ((n3c_{best} - n3c_b)n3c_{best})]$$

in which $n3c_{bes_t}$ is the current biggest number of 3-cliques, which has been obtained by the ants since the beginning of the algorithm operation up to this point, $n3c_b$ is the biggest number of 3-cliques, which has been obtained by all ants in one cycle.

When both loops have been terminated, the best solution is returned. At the beginning of each inner loop a start point is prepared for each ant. From such a start point each ant begins to create a solution to the optimization problem and then, in the *while* loop, each ant selects a next element j from the set M with the probability $\mathrm{p}(j)$ and puts it into its solution set S. The probability $\mathrm{p}(j)$ can be expressed by the formula

$$p(j) = \frac{t_j n_j}{\sum_j (t_j n_j)} \tag{1}$$

where t_j– is the quantity of the pheromone deposited on element j ($1 \leq j \leq$ max), "max" being the maximum number of available elements, from which a selection can be made, from the set M at the start and then from the set A

in the loop *while*, n_j is a parameter, expressing the desirability of including an element j from the set A into the solution the set S.

Algorithm 1 The General Ant Algorithm

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

4. The proposed ant algorithm

The structure of the proposed algorithm can be divided into two parts. In the first part ants are looking for the maximum matching in a bi-partite graph. Ants use the pheromone and a heuristic to find the optimal solution, that is – the maximal matching in a bi-partite graph. The heuristics they use concerns the number of vertices from the third part of a 3-partite graph. This means that this heuristics lets us to find not only the maximum matching in a bi-partite graph, but also such a maximum matching in a bi-partite graph, which has the maximum number of 3-vertices cliques that can be created with vertices from the third part of a 3-partite graph.

The bi-partite graph is a subgraph of the three-partite graph. The maximum matching in a bi-partite graph consists of edges, which connect vertices v_i from the set V_i and v_j from the set V_j. Such an edge (v_i, v_j) can belong to different 3-cliques, which means that such two vertices (an edge) can, together with some vertices form the third part of the given graph, constitute different 3-cliques and the heuristic in question concerns the number of such possible 3-cliques, so that when we have two maximal matchings in a bi-partite graph with an equal number of edges, this heuristics lets us obtain this maximal matching, which has the greater number of the possible 3-cliques.

Next, in the second part of the ant algorithm this maximal matching in a bipartite graph is used by ants together with only the pheromone-based information to find the maximum number of 3-vertices cliques in a 3-partite graph.

All this creates the ant algorithm with heuristics in its first part, this heuristics having been presented in Schiff (2020), while the ant algorithm for this problem yet without the heuristics was presented in Schiff (2018). A new feature of the ant algorithm, as it is proposed here, is the heuristics, which is included in the second part of the ant algorithm, and this is the aspect, by which the present algorithm differs from that, which has been discussed in Schiff (2020), and to which we shall refer to here as ACO1. Now, in the second part of the algorithm ants use not only the pheromone-based information, but also a heuristics in order to find the maximum number of 3-vertices cliques in a 3-partite graph, and this algorithm will be further called ACO2. This heuristics is expressed as follows:

$$N(v_i) = 1/lz_i \tag{2}$$

where lz_i is the number of 3-vertices cliques, in which the given vertex v_i could be included, this vertex v_i belonging to the third part of a 3-partite graph, not considered in the first phase of the algorithm. In the first phase of the algorithm the maximum matching in bi-partite graph is obtained. The bi-partite graph is the subgraph of the three-partite graph. Each of the vertices in the third part of the 3-partite graph can, together with different edges from the maximum matching in the bi-partite graph, constitute different 3-cliques, so to each vertex v_i from the third part of the 3-partite graph we can assign a number lz_i. The number lz_i is the number of 3-cliques, which can be constituted by a vertex (here: vertex v_i) from the third part of the 3-partite graph with the edges from the maximum matching in the bi-partite graph, obtained in the first phase of the algorithm.

The function $N(v_i)$ is the desirability function, which is used in the second phase of the ant algorithm, analogous to the coefficient n_j, appearing in formula (1), so that each ant in the second phase of the algorithm selects a vertex v_k from the third part of 3-partite graph and assigns it to the edge from maximum matching in the bi-partite graph, which has been obtained in the first phase of the ant algorithm, and thus each ant builds 3-vertices cliques, using the value of this desirability function. The values of coefficients n_j and $N(v_i)$ are calculated during the functioning of the algorithm and they reflect the degree of appropriateness of the element j or the vertex v_i in the perspective of obtaining the best solution. The higher the value of the desirability function, the more useful the element j or the vertex v_i is in getting the best solution.

5. Results of experiments

The first experiment concerns the average of the maximum number of 3-vertices cliques obtained using the ACO1 and ACO2 algorithms for different sizes of the problem, namely $n = \{50, 100, 150, 200, 250\}$. The ACO2 is the ant algorithm with the heuristics, presented in this paper and the ACO1 algorithm is the ant algorithm, which is presented in Schiff (2020). The number n is the number of vertices in any part of the 3-partite graph, meaning that there are the same numbers of vertices in each part of the 3-partite graph. The average number of 3-vertices cliques is calculated from 10 experimental measurements and they are shown in Table 1 and in Fig. 2. The parameter r is the evaporation rate, the parameter lc is the number of cycles, the parameter lm is the number of ants and the parameter q is the density (interpreted as probability of edge existence).

Table 1. Average numbers of cliques for different n, r =0.998, lc=20, lm=30, q =0.05

n	50	100	150	200	250
ACO1	10.4	52.4	109.7	166.1	222.1
ACO2	10.4	52.9	114.0	172.6	227.1

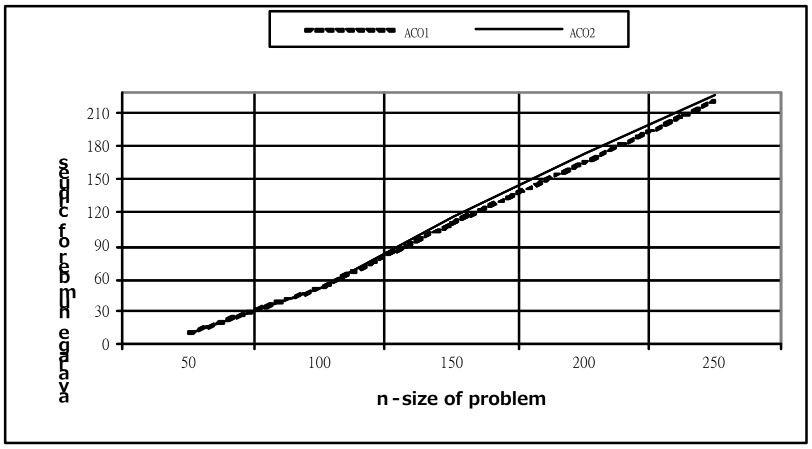

Figure 2. Average numbers of cliques for different n, r =0.998, lc=20, lm=30, q =0.05

The next experiment was conducted for the constant size of the problem but for variable evaporation rates, r ={0.99, 0.992, 0.994, 0.996, 0.998}, and its results are shown in Table 2 and in Fig. 3. Then, the subsequent experiment consisted in changing the graph density, with q ={0.01, 0.03, 0.05, 0.07, 0.09}, and its results are shown in Table 3 and in Fig. 4.

Two last experiments concerned the changes in such algorithmic parameters as the number of cycles, lc={20, 30, 40, 50}, and the number of ants, lm={20, 30, 40, 50}, and the respective results are shown in Tables 4 and 5, and in Figs. 5 and 6. In all of these experiments we can see that the ACO2 algorithm performs better than ACO1.

Table 2. Average numbers of cliques for different evaporation rates r, q =0.05, n =200, lc=20, lm=30

r	0.998	0.996	0.994	0.992	0.990
ACO1	166.1	166.4	166.2	167.6	166.9
ACO2	172.6	171.6	171.3	172.0	172.5

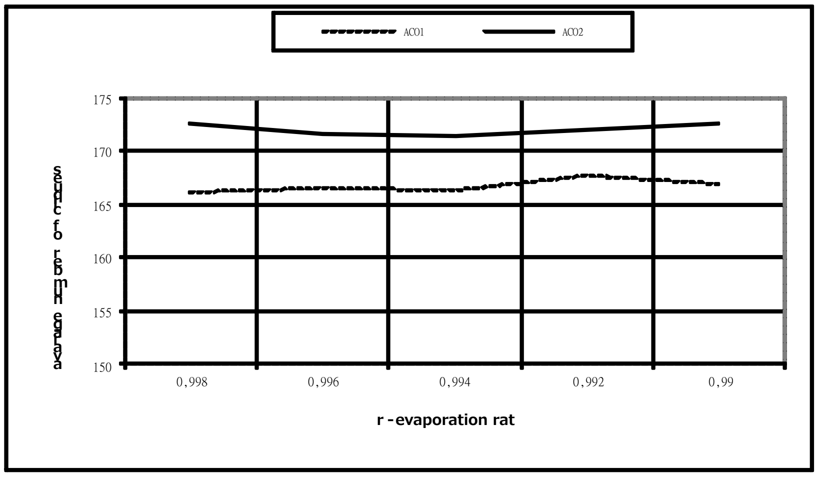

Figure 3. Average numbers of cliques for different evaporation rates r, q =0.05, n =200, lc=20, lm=30

Table 3. Average numbers of cliques for different densities q, r =0.998, n =200, lc=20, lm=30

q	0.09	0.07	0.05	0.03	0.01
ACO1	193.2	187.0	166.1	100.0	57.0
ACO2	193.8	189.0	172.6	101.6	57.0

Table 4. Average numbers of cliques for different numbers of cycles in the algorithm, lc, with q =0.05, r =0.998, n =200, lm=30

lc	20	30	40	50
ACO1	166.1	167.3	167.7	169.5
ACO2	172.6	172.4	173.4	175.1

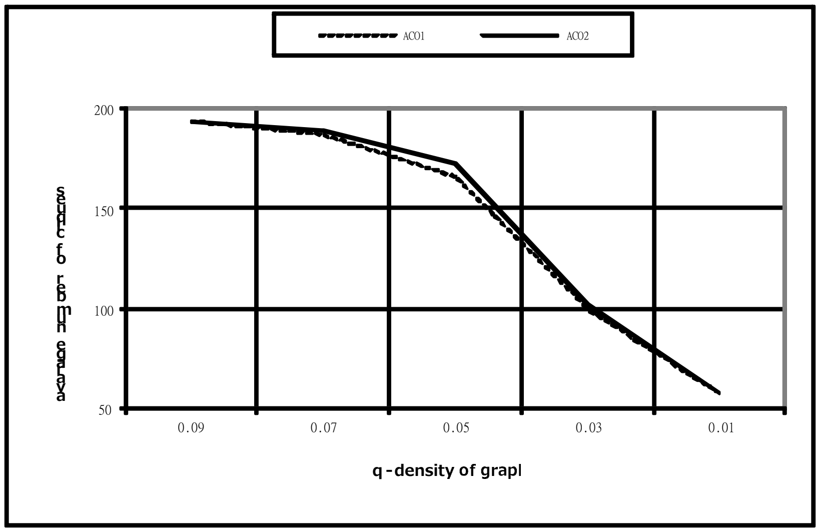

Figure 4. Average numbers of cliques for different values of lm, for n =250, lc=200, r =0.997

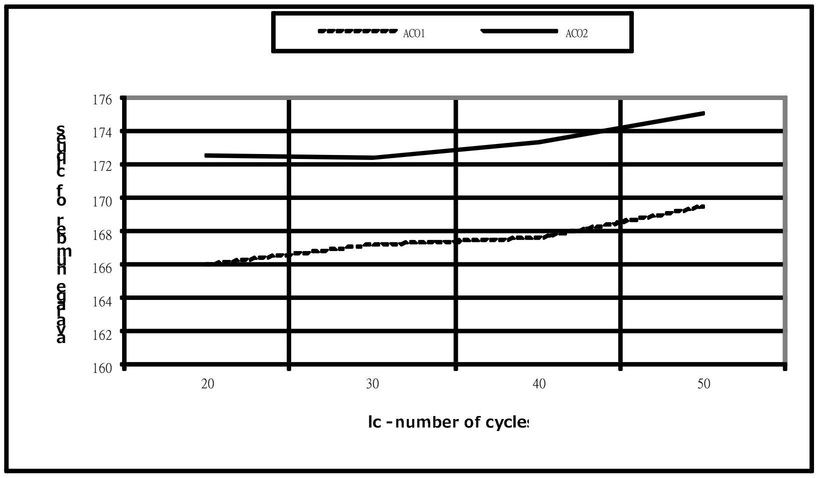

Figure 5. Average numbers of cliques for different numbers of cycles, lc, with q =0.05, r =0.998, n =200, lm=30

Table 5. Average numbers of cliques for different numbers of ants, lm, with q =0.05, r =0.998, n =200, lc=20

Lm	20	30	40	50
ACO1	166.1	166.1	167.0	165.7
ACO2	171.3	172.6	171.8	171.7

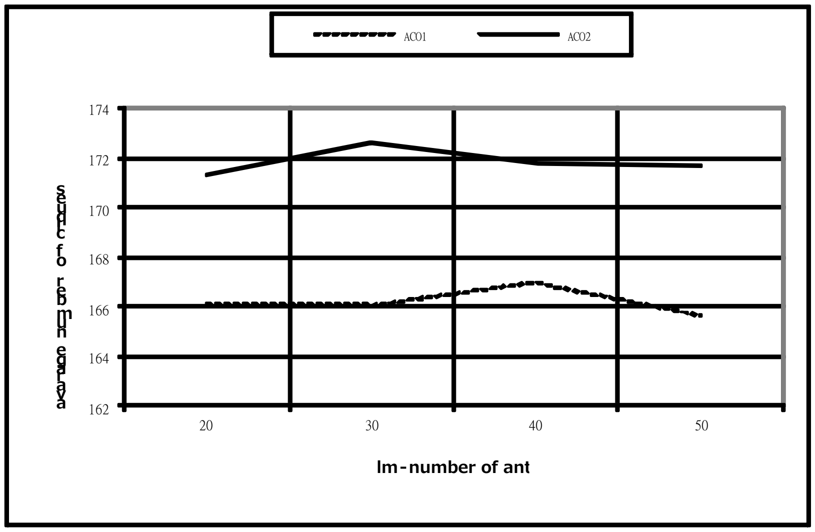

Figure 6. Average numbers of cliques for different numbers of ants, lm, with q =0.05, r =0.998, n =20, lc=20

Finally, we present below the results of comparison of the ACO1 algorithm with the ACO algorithm (the ant algorithm without any desirability function introduced, see Schiff, 2018) in Table 6 for different numbers of vertices, equal in each part of the 3-partite graph, and the results of comparison of the ACO algorithm with the Apx3Dmatch-F algorithm from Chen (2012), which are shown in Table 7.

Table 6. Average number of 3-cliques obtained for lc=100, lm=30, r =0.998 and q =0.07. Comparison of ACO and ACO1 algorithms

n	20	30	40	50	60
ACO	1.1	6.3	11.2	17.9	29.5
ACO1	1.1	6.7	12.2	19.9	32.2

Table 7. Average number of 3-cliques obtained for lc=100, lm=30, r =0.998 and q =0.07. Comparison of ACo and Apx3Dmatch-F algorithms

n	10	20	30	40	50
ACO	0.5	3.3	6.7	12.1	29.9
Apx3Dmatch-F	0.5	3.3	6.5	11.4	17.7

As we can see from Table 7, the ACO ant algorithm without any desirability function performs better than the approximation algorithm (Apx3Dmatch-F), while Table 6 shows that the ACO1 ant algorithm with desirability function, which is used in the first phase of the ant algorithm, performs better than the ACO ant algorithm without any desirability function.

6. Conclusion

Experiments, carried out and reported here, have shown that the newly proposed ACO2 algorithm has a clear and persistent superiority over the ACO1 algorithm for graphs with different graph density and for different sizes of the problem as regards the number of cliques found. This new ACO2 algorithm can be easily adapted to finding the maximum number of, more generally, d-vertices cliques in a d-partite graph. In the ACO1 algorithm the selection probability formula has a classical form with desirability pattern and is applied in one, first phase of the algorithm, while in the ACO2 algorithm the selection probability formula has again a classical form with desirability pattern, but it is applied in both phases of the algorithm.

The maximum matching in the bi-partite graph is being obtained in the first phase of the ant algorithm. This maximum matching contains the set of edges

$\{e_{ij}\}$. In this first phase a desirability function, which is used in the algorithm, is described as follows: first we calculate for each edge e_{ij} between two parts of the 3-partite graph the number of 3-cliques, *l3k*, i.e. *l3k* is the number of vertices from the third part of the 3-partite graph and these are the vertices, which constitute together with the edge e_{ij} the 3-cliques we look for. Thus, each edge e_{ij} is being assigned a number *l3k*. The number *l3kmax* is the biggest number among the *l3k*, i.e. among those that have been assigned to the edges of the maximum matching in bi-partite graph. Now, the desirability function in the first phase of the ant algorithm is represented through the formula

$N(e_{ij}\ ) = 1/[1 - (l3kmax - l3k)/l3k]$.

This formula is used in the first phase of ACO2 and ACO1 algorithms. In the second phase of the ACO1 algorithm there is no desirability formula, while in the second phase of the ACO2 algorithm the desirability formula is used as presented in this paper, as expressed through formula (2).

References

Biro, P. and McDemid, E. (2010) Three-sided stable matching with cyclic preferences. *Algorithmica*, **58**, 1, 5-18.

Chen, J. (2012) Iterative expansion and Color Coding: an Improved Algorithm for 3D-Matching. *ACM Transactions on Algorithms*, 6.1-6.22.

Dorigo, M., Di Caro, G. and Gambardella, L. M. (1999) Ant algorithms for discrete optimization. *Artificial Life*, **5**, 2, 137–172.

Eriksson, K., Sjöstrand, J. and Strimling, P. (2006) Three-dimensional stable matching with cyclic preferences. *Mathematical Social Sciences*, **52**, 1, 77-87.

Karp, R.M. (1972) Reducibility among Combinatorial Problems. In: R. E. Miller and J. W. Thatcher, eds., *Complexity of Computer Computation.* Plenum Press, N.Y., 85-103.

Knuth, D. (1997) Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms. *Amer. Math. Soc.*, Providence, RI.

Schiff, K. (2018) An ant algorithm for the triple matching problem. *Technical Transactions*, *Electrical Engineering*, **115**, 2, 179-186.

Schiff, K. (2020) An improved ant algorithm for the triple matching problem. *Technical Transactions*, **13**, 1, art no. 20200005, 1-7.