TECHNICAL TRANSACTIONS | CZASOPISMO TECHNICZNE
AUTOMATIC CONTROL | AUTOMATYKA

4-AC/2013

KRZYSZTOF SCHIFF[*]

# ANT COLONY OPIMIZATION ALGORITHMS FOR CLUSTERING PROBLEMS

## ALGORYTMY MRÓWKOWE DLA PROBLEMÓW KLASTERYZACJI

A b s t r a c t

The clustering problem is one of the main problems which can be encountered in a data analysis. This problem can be modelled by means of a graph; finding clusters means finding cliques in the graph. Often there is a need to find clusters (cliques) in a graph in different ways and to construct a list of clusters. This paper describes two such ways, these can be stated as the cluster minimum covering problem and the vertex cluster minimum partitioning problem. This paper describes new ant algorithms which were used in order to make a list of clusters in both presented problems, and also discusses the results of their comparison.

*Keywords*: *clustering, clique covering problem, clique vertex partitioning problem, ant algorithms*

S t r e s z c z e n i e

Problem klasteryzacji jest jednym z często spotykanych problemów w analizie danych. Problem klasteryzacji może być zamodelowany przy pomocy grafów i znajdowanie klasterów sprowadza się wówczas do znajdowania klik w grafach. W tym artykule opisano dwa sposoby wyznaczania klasterów, czyli klik w grafach, takich jak: problem pokrycia klastrami (klikami) grafu oraz problem wierzchołkowego podziału grafu na klastry (kliki) oraz także przedstawiono dwa nowe algorytmy bazujące na zachowaniu mrówek służące do wyznaczania klastrów (klik) dla obu problemów, a także dokonano porównania ich ze znanymi algorytmami rozwiązującymi te problemy.

*Słowa kluczowe*: *klasteryzacja, pokrycie klikami, wierzchołkowy podział na kliki, algorytm mrów*

[*] Ph.D. Krzysztof Schiff, e-mail: kschiff@pk.edu.pl, Department of Automatic Control and Information Technology, Faculty of Electrical and Computer Engineering, Cracow University of Technology.

## 1. Introduction

The clustering problem can be encountered in many optimisation problems, which are often difficult – finding solutions to such problems takes a great deal of time. Clustering techniques are used to shorten the time needed to find solutions to these problems. Data and dependency between data can be modelled by means of a graph. These data can be grouped into clusters according to the dependency of their characteristics. Finding clusters means finding cliques in a graph. The clique covering problem and the vertex clique partitioning problem are NP-difficult problems [1, 16]. Many papers have been devoted to clique problems [6, 8–12, 18]. Ant algorithms have been used to find maximum cliques or to find all cliques in a graph [2–5, 15]. The clique covering problem and the vertex clique partitioning problem were solved by means of a neural network [7, 13] and a genetic algorithm [14, 17]. This paper describes two new ant algorithms for the problems in question.

## 2. Models for cluster analysis

Let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices in graph $G$ and $E = \{e_1, e_2, ..., e_m\}$ is a set of edges $m \leq n^2$. Graph $G = (V, E)$ is complete if each pair of its vertices $v_i, v_j \subseteq V$ is connected by edge $e_{ij} = \{v_i, v_j\} \subseteq E$. The clique $C$ is a subset of all vertices which constitute the set $V$ and which, when combined, constitute a complete graph. Clique $C$ is called the cluster. Clique $C$ is maximal if it is not included in another clique. It is maximum if this is a maximal clique and there is any other maximal clique in the graph with a higher number of vertices than this maximum clique. Relations between objects and their features and relations between features of different objects can be modelled by graphs or matrices. Vertices can represent only objects or objects and features. Edges can represent the existence of a common feature between two objects or the possession of a feature; this is shown in Fig. 1a and in Fig. 1b, respectively. A value of 1 in each matrix represents an existence and a value of 0 represents an absence.

a)

| c\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

b)

| c\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

Fig. 1. Matrices for cluster models

An object is represented by variable $n$ in Fig. 1a. If there is a feature common to two objects, a value of 1 is used, and if there is no common feature, a value 0 of is used in the matrix. The cluster which consists of objects {2, 3, 4, 5} means that objects 2, 3, 4 and 5 possess the same feature. If there is a need to investigate many features between many objects, then for each of these features, a graph or matrix model should be built and for each of these features in each of these models, a cluster analysis should be performed. This means that a list of maximal cliques should be indicated.

An object is represented by variable *n* and a feature is represented by variable *c* in Fig. 1b. Both are represented by a graph vertex, matrix column or matrix verse. If an object possesses a feature *c,* a value of 1 is used, and if it does not possess feature *c*, a value of 0 is used in the matrix at the crossing of the matrix column and matrix verse. A cluster which consists of objects {1, 2, 3, 4} and features {1, 2, 3, 4} means that objects 1, 2, 3, 4 possess common features 1, 2, 3 and 4. The graph shown in Fig. 1b can be used only in cases where the number of objects and features is equal. If the number of objects and features is not equal, these numbers have to be made equal. Either the verses of the matrix should be filled with a value of 1 in cases where the number of objects is higher than the number of features, as shown in Fig. 2a, or columns of the matrix should be filled with a value of 1 in the opposite case, as shown in Fig. 2b; thus, both of these cases are transformed to the model shown in Fig. 1b.

A case in which the number of features is less than the number of objects is shown in Fig. 2a. There are 5 features and 6 objects, so the 6th verse is filled with a value of 1; this means that a common feature was added to all objects. A case in which the number of features is greater than the number objects is shown in Fig. 2b. There are 6 features and 5 objects, so the 6th column is filled with a value of 1; this means that an object was augmented with features which are common to other objects. After the maximum clique has been indicated, the added features or added objects are removed.

a)

| c\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 |

b)

| c\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 0 | 1 | 1 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

Fig. 2. Matrix: a) when the number of features is lower than the number of objects, b) in the opposite case

If there is a need to perform a multi-feature cluster analysis, then for each of these features, a cluster matrix and later, a cumulative matrix are created. Matrices for 4 different features of 6 objects are shown in Fig. 3a, b, c and d. There is matrix of 6 columns by 6 verses. Cumulative matrices for a probability equal to 0.6 are shown in Fig. 4a; a probability equal to 0.8 is shown in Fig. 4b. Matrices 4a and 4b have been obtained such that the number of features for each object in all 4 matrices shown in Fig. 3a, b, c and d were added and divided by the number of matrices. Thus the obtained average values of feature existence were compared with the value of probability and a)

| n\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

b)

| n\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

c)

| n\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

d)

| n\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 2 | 0 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 1 | 0 | 1 |

Fig. 3. Matrices – for features a, b, c and d

a)

| n\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

b)

| n\n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 1 |

Fig. 4. Cumulative matrices

then, if the value of probability was lower than the average value of the existence of features, a value of 1 was written as needed in the cumulative matrix. For example: there is a cellule at the crossing of verse 2 and column 1 in all 4 matrices; in this cellule, a value of 1 is encountered 3 times in all 4 matrices. If the number 3 is divided by 4 (since there were 4 matrices), then the number 0.75 is obtained. When the probability value is equal to 0.6, then in this cellule, a value of 1 should be written as in Fig. 4a, but when the probability value is equal to 0.8, then in this cellule, a value of 0 should be written as in Fig. 4b. A cumulative matrix for cluster analysis with a probability of 0.6 is shown in Fig. 4a and with a probability of 0.8, is shown in Fig. 4b.

## 3. Two cluster problems

Set $S = \{C_1, C_2, ..., C_k\}$ is a cover of the graph $G$, if all graph edges $E(G)$ are covered by edges of cliques $C_i$ and:

$$E(G) = \sum_{i=1}^{k} E(C_i) \tag{1}$$

Where: $k$ is the number of cliques (clusters).

If there is a set $S$ of clusters in which each edge belongs to at least one cluster $C_i$, $1 \le i \le k$, then $S$ is a graph covered by clusters. The cardinality number of $S$ is called the cluster covering number of graph $G$ and is marked by cc($G$).

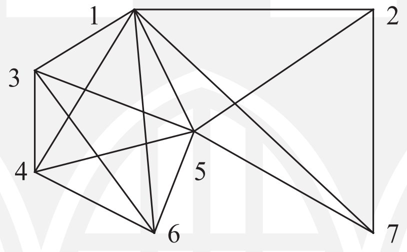

Fig. 5. A graph for two problems

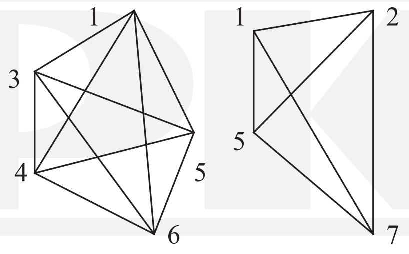

Fig. 6. A solution to the clique covering problem

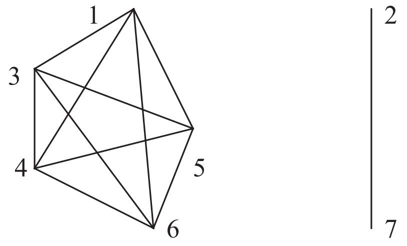

Fig. 7. A solution to the vertex clique partitioning problem

A set $S$ of clusters in which each vertex belongs to exactly one element of $S$ means that for two different cliques $C_i$ and $C_j$, $1 \le i \le k$, $1 \le j \le k$, $i \ne j$, there are no common vertices. $V(C_i) \cap V(C_j) = \varnothing$ is called a graph vertex partitioning on clusters. The cardinality number of $S$ is called the clique vertex partitioning number of graph $G$ and is marked by cwp($G$).

Definition: minimum clique (cluster) covering problem in a graph

A minimum number of maximal cliques is searched for in graph $G = (V, E)$ such that each graph edge has two of its ends in at least one clique; in other words, a set S with a minimum number cc($G$) is searched for.

Definition: minimum clique (cluster) vertex partitioning problem in graph

A minimum number of maximal cliques is searched for in graph $G = (V, E)$ such that each graph vertex belongs to exactly one clique; in other words, a set S with minimum number cwp($G$) is searched for.

Differences between these two problems were shown in Figs. 6 and 7 as a different kind of solution for an example of the graph presented in Fig. 5. This means a solution to the clique covering problem and a solution to the vertex clique partitioning problem, respectively.

## 4. Ant algorithms for cluster problems

The solution to the minimum cluster covering problem is a list of maximal cliques. There should be no repetitions within the list. Each of these maximal cliques is constructed by ants for each graph edge. The pseudo-code of the ant algorithm is shown as algorithm 1.

The solution for the minimum vertex partitioning problem is a list of maximal cliques. Each of these maximal cliques is constructed by ants in a current graph structure, which is obtained from the preceding graph structure by removing its edges and vertices. The pseudo-code of the ant algorithm was shown as algorithm 2.

Maximal cliques were determined in both algorithms. In general, a maximal clique is created by vertices which are neighbours, this means that each pair of vertices from a clique are connected by an edge. A maximal clique is created so that at first, one and then another vertex is selected from the neighbours of the first selected vertex; then the third vertex is selected from the neighbours of the first two selected vertices and so on, until there is no vertex among the neighbours of the vertices already selected. The order in which these vertices are selected influences the size of the created maximal clique. The size of the clique depends on the vertex selection sequence; thus it is important to know which vertices should be selected, and in which sequence, in order to obtain the maximal clique. Since there are so many possible selection sequences, there is no way to check them all out; this is why ant algorithms are used to select vertices and to create the maximal clique.

Algorithm 1

**Cluster covering procedure**

```
Repeat for each graph edge
   while (there is a cycle to repeat)
            while (there is an ant which has not yet worked)
          while (the clique has not been completed)
                include one of the vertices next to the maximal clique with probability $p$
       remember the best solution which was found by all ants in one cycle
   remember the best solution which has been found so far in all cycles
   update pheromone trails
   include the clique on the list
```

Algorithm 2

**Cluster vertex partitioning procedure**

```
while (there is a cycle to repeat)
        while (there is an ant which has not yet worked)
            while (the clique has not been completed)
              include one of the vertices next to the maximal clique with probability $p$
        remember the best solution which was found by all ants in one cycle
  remember the best solution which has been found so far in all cycles
  update pheromone trails
  remove vertices and edges which participated in the last clique from the graph
  include the clique on the list
```

Algorithm 3

**Maximum clique (cluster) procedure**

```
$C = \varnothing$
$N = \varnothing$
select a first vertex $v_i \in V$
$C = C + \{v_i\}$
$N = N +$ all $\{v_j : (v_j, v_i) \in E\}$
**while** $N \neq \varnothing$ **do**
  select a vertex $v_i \in N$ with probability $p(i)$
  $C = C + \{v_i\}$
  $N = N +$ all $\{v_j : (v_i, v_j) \in E\}$
**end while**
return $C$
```

In the ant algorithm, which would be called the ALG algorithm later in this paper and which is presented in paper [4], a vertex is included in the maximal cluster with probability:

$$p(j) = \frac{[t_j]^x}{\sum_{v_j \varepsilon \text{Candidates}} [t_j]^x} \tag{2}$$

In the ant algorithm, which would be called the NALG algorithm later in this paper and which is presented in this paper, a vertex is included in the maximal cluster with probability:

$$p_j = \begin{cases} \dfrac{\tau_j^\alpha n_j^\beta}{\sum_{j \in N_i} \tau_j^\alpha n_j^\beta}, & \text{for } \; j \in N_i \\ 0, & \text{for } \; j \text{ which not} \notin N_i \end{cases} \tag{3}$$

This probability depends on the pheromone trail and on the desire for vertex selection expressed by the formula:

$$n_j = \frac{d_j d_j}{\sum_{j=1}^{m} (d_j d_j)}, \;\; j \in N_i \tag{4}$$

where: $d_j$ is a vertex degree, i.e., the number of edges adjacent to vertex $j$.

After all ants have worked in each cycle, some of the pheromone evaporates at rate $r$ according to expression $\tau = r\tau$. On all vertices which, taken together, constitute the maximal clique, a pheromone is deposited – this pheromone quantity is expressed as

$$\Delta\tau = \frac{1}{1 + \dfrac{cs_{best} - cs}{cs_{best}}} \tag{4.4}$$

where: $cs$ is the size of the maximal clique.

## 5. Experiments

Tests were conducted for both of the ant algorithms, ALG and NALG, mentioned above and for two problems which have been taken into consideration: the minimum clique covering problem and the minimum vertex partitioning problem.

The first test was conducted for the minimum clique covering problem for a constant number of vertices $n = 100$ and for different graph densities $q = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ with a constant evaporation rate $r = 0.99$, a constant number of cycles $lc = 50$ and a constant number of ants $m$ = 30. Average values from 100 measurements are shown in Table 1. It is advantageous to use the NALG algorithm rather than the ALG, since there is a positive difference over the entire range of graph density {0.1, 0.3, 0.5, 0.7, 0.9}.

Table 1

**Covering: number of cliques as a function of graph density**

| q | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| ALG | 306.2 | 948.9 | 2184.8 | 3402.8 | 4450.0 |
| NALG | 306.0 | 761.4 | 1975.1 | 3369.3 | 4449.7 |
| ALG – NALG | 0.2 | 187.5 | 209.7 | 33.5 | 0.3 |

The second test was also conducted for the minimum clique covering problem, but for a different number of vertices $n$ = {50, 100, 150, 200, 250} and for a constant graph density $q$ = 0.5 with a constant evaporation rate $r$ = 0.99, a constant number of cycles $lc$ = 50 and a constant number of ants $m$ = 30. Average values from 100 measurements are shown in Table 2. It is advantageous to use the NALG algorithm rather than the ALG, since there is a positive difference over the entire range of graph vertices {50, 100, 150, 200, 250}.

Table 2

**Covering: number of cliques as a function of the number of vertices**

| n | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| ALG | 390.9 | 1975.1 | 5150.5 | 9446.8 | 15489 |
| NALG | 457.5 | 2184.8 | 5397.5 | 9688.2 | 15654.8 |
| ALG – NALG | 66.6 | 209.7 | 247 | 241.4 | 165.8 |

The third test was conducted for the minimum vertex clique partitioning problem, for a constant number of vertices $n$ = 100 and for a different graph density $q$ = {0.1, 0.3, 0.5, 0.7, 0.9} with a constant evaporation rate $r$ = 0.99, a constant number of cycles $lc$ = 50 and a constant number of ants $m$ = 30. Average values from 100 measurements are shown in Table 3. It is advantageous to use the NALG algorithm rather than the ALG, since there is a positive difference over the entire range of graph density {0.1, 0.3, 0.5, 0.7, 0.9}.

Table 3

**Vertex partitioning: number of cliques as a function of graph density**

| q | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| ALG | 42.58 | 27.1 | 17.78 | 12.53 | 6.8 |
| NALG | 42.47 | 26.8 | 17.61 | 12.45 | 6.69 |
| ALG – NALG | 0.11 | 0.3 | 0.17 | 0.08 | 0.11 |

The last test was conducted for the minimum vertex clique partitioning problem, but for a different number of vertices *n* = {50, 100, 150, 200, 250} and for a constant graph density *q* = 0.5 with a constant evaporation rate *r* = 0.99, a constant number of cycles *lc* = 50 and a constant number of ants *m* = 30. Average values from 100 measurements are shown in Table 4. It is advantageous to use the NALG algorithm rather than the ALG, since there is a positive difference over the entire range of graph vertices {50, 100, 150, 200, 250}.

Table 4

**Vertex partitioning: number of cliques as a function of the number of vertices**

| n | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| ALG | 11.95 | 17.78 | 24.44 | 30.2 | 35.33 |
| NALG | 11.91 | 17.61 | 24.11 | 29.76 | 34.52 |
| ALG – NALG | 0.04 | 0.17 | 0.33 | 0.44 | 0.81 |

## 6. Conclusion

The NALG algorithm described in this paper was compared with the previously elaborated ALG algorithm. It has been shown that the NALG algorithm has a permanent advantage over the ALG, since it obtained lists of clusters in both problems taken into consideration shorter than those obtained by the already elaborated ALG for a broad range of graph density and number of graph vertices.

## References

[1] Garey M., Johnson D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York 1979.

[2] Ponce J., Ponce E., Padilla F., Padilla A., *Ant Colony Algorithm for Clustering through of Cliques*, 2006.

[3] Ponce J., Ponce E., Padilla F., Padilla A., Ochoa A., *Algoritmo De Colonia De Hormigas Para El Problema Del Clique Máximo Con Un Optimizador Local K-Opt*, Hífen, Uruguaiana, Vol. 30, No. 58, Brasil 2006.

[4] Fenet S., Solnon C., *Searching for Maximum Cliques with Ant Colony Optimization EvoWorkshops 2003*, LNCS 2611, 2003, 236-245.

[5] Xu X., Ma J., Lee J., *An Improved Ant Colony Optimization for the Maximum Clique Problem*, IEEE Third International Conference on Natural Computation, 2007.

[6] Bron C., Kerbosh J., *Finding all cliques of an undirected graph*, Association of Computer Machine, 16, 1973, 575-577.

[7] Wang J., Xu X., Tsng Z., Bi W., Chen X., Li Y., *A New Neural Network Algorithm for Clique Vertex-Partition Problem*, Yin F., Wang J., Guo C. (Ed.): ISNN 2004, LNCS 3173, Springer-Verlag, Berlin 2004, 425-29.

[8] Kellerman E., *Determination of keyword conflict*, IBM Technical Disclosure Bulletin 16, 1973, 544-546.

[9] Behrisch M., Taraz A., *Efficiently covering complex networks with cliques of similar vertices*, TCS: Theory of Computer Science 355, 2006, 37-47.

[10] Cazals F., Karnade C., *An algorithm for reporting maximal c-cliques*, Theory of Computer Science 349, 2005, 484-490.

[11] Gramm J., Guo J., Huffner F., Niedermeier R., *Data reduction and exact algorithms for clique cover*, ACM J. Exp. Algorith. 13, 2.2–2.15, 2009.

[12] Tseng C., Siewiorek D., IEEE Trans. CAD 5, 379 (1986).

[13] Harmanani H., *A Parallel Neural Networks Algorithm for the Clique Partitioning Problem*, IJCA, Vol. 9, No. 2, 2002

[14] Little P., Rylander B., *Problem Partitioning in Hybrid Genetic Algorithms*, Proceedings of the 5th WSEAS Int. Conf. on circuits, systems, electronics, control and signal processing, Dallas, USA, November 1–3, 2006

[15] Rizzo J., *An Ant System Algorithm for Maximum Clique*, The Pennsylvania State University, Master Thesis, 2003.

[16] Orlin J., *Contentment in graph theory: covering graphs with cliques*, Mathematics 39, 1977, 406-424.

[17] Snyers D., Clique Partitioning Problem and Genetic Algorithms in Albrecht R.F.(eds.), Artificial Neural Nets and Genetic Algorithms Springer-Verlag, 1993.

[18] Tseng C., Sieworek D., *Facet: A Procedure for the Automated Synthesis of Digital Systems*, Proceeding of the 20th ACM/IEEE Design Automated Conf., 1983, 490-496.