Evolutionary Algorithm to Find Graph Covering Subsets Using α-Cliques

Henryk Potrzebowski, Jarosław Stańczak, Krzysztof Sęp

Systems Research Institute, Polish Academy of Science, Warsaw, Poland
e-mail: potrzeb@ibspan.waw.pl
stanczak@ibspan.waw.pl
sep@ibspan.waw.pl

Abstract. The article describes a new evolutionary based method to divide graph into strongly connected structures we called α-cliques. The α-clique is a generalization of a clique concept with the introduction of parameter α. Using this parameter it is possible to control the degree (or strength) of connections among vertices (nodes) of this sub-graph structure. The evolutionary approach is proposed as a method that enables to find separate α-cliques that cover the set of graph vertices.

1. Basic Concepts

A *graph* is a pair $G=(V, E)$, where V is a non-empty set of vertices and E is a set of edges. Each edge is a pair of vertices $\{v_1, v_2\}$ that $v_1 \neq v_2$. [2,15]

A clique $Q=(V_q, E_q)$ in graph $G=(V, E)$ is a graph that $V_q \subseteq V$ and $E_q \subseteq E$ where each pair $v_1, v_2 \in V_q$ of vertices fulfills a condition $\{v_1, v_2\} \in E_q$. [4,11]

Sub-graph of $G=(V, E)$ is a graph $G'=(V', E')$ where $V' \subseteq V$ and $E' \subseteq E$ that for all $e \in E$ and $e=\{v_1,v_2\}$ if $v_1,v_2 \in V'$ than $e \in E'$ [15].

2. The Concept of α-Clique

In many cases there is no need (or is not possible) to find an appropriate family of sets where the elements within each set of that family are fully connected with other elements of the same set. It is often sufficient to find a family of sets where its elements are connected with at least a predefined percentage of elements from this set. Thus we define a concept of an α-clique to introduce a sub-graph with desired properties.

The α-clique in a graph $G = (V, E)$ is a sub-graph $Q_\alpha = (V_\alpha, E_\alpha)$ that $V_\alpha \subseteq V$ and $E_\alpha \subseteq E$, $k=Card(V_\alpha)$ for all $v_i \in V_\alpha$, k_i is a number of vertices $v_j \in V_\alpha$ that $\{v_i, v_j\} \in E_\alpha$ that $\alpha \leq k_\alpha / k$.

The structure defined above possesses some interesting properties:

–it is very simple to prove that if $\alpha > 0.5$ then α-clique is a connected graph;
–single vertex forms the smallest possible α-clique;
–not every subset of α-clique forms α-clique with desired value of α, (this property is illustrated by the following example).

For example: for α=0.8 we have α-clique as it is shown in the left part of Figure 1. (for all vertices $\alpha = k_\alpha / k$).

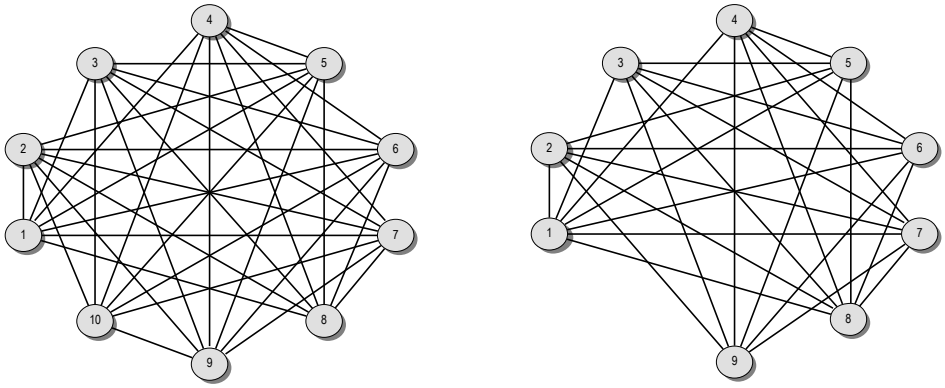

Figure 1. An example of α-clique (α=0.8) – left and a sub-graph, which is not such α-clique – right.

If we remove some node (all nodes have degree equal 8) we acquire a graph as it is shown in the right part of Figure 1. The degree of the node number 9 is 7, but 7/9<0.8 so this graph is not an α-clique with desired value of α equal 0.8, but of course it is an example of α-clique with α=0.7 or smaller.

3. Evolutionary Approach to Find α-Cliques

The standard evolutionary algorithm (EA) works in the manner as it is shown in the Algorithm 1, but this simple scheme requires many problem specific improvements to work efficiently.

The adjustment of the genetic algorithm to the problem requires a proper encoding of solutions, an invention of specialized genetic operators for the problem, accepted data structure and a fitness function to be optimized by the algorithm.

1. Random initialization of the population of solutions.
2. Reproduction and modification of solutions using genetic operators.
3. Evaluation of obtained solutions.
4. Selection of individuals for the next generation.
5. If stop condition is not satisfied, go to 2.

Algorithm 1. The evolutionary algorithm.

3.1 Individual Representation

The whole information about the problem is stored in an array of data that describes all data connections. This array can be binary (a matrix of incidence of undirected graph: 0 – no connection, 1- presence of connection) or non-negative (undirected graph) real-valued and in this case the stored value denotes the strength of the connection.

Members of the population (Figure 2) contain their own solutions of the problem as a dynamic table of derived α-cliques (number of them may change during computations). Each

element of this table (α-clique) has a list of nodes attached to this clique and each node is considered only once in one solution (population member). Unattached nodes are also included, they constitute small, one-element α-cliques (one node is also α-clique with α=1). Thus each solution contains all nodes from a graph described by incidence matrix. But the solution with many small α-cliques is rather not profitable, and it is the role of evolutionary algorithm to find bigger ones.

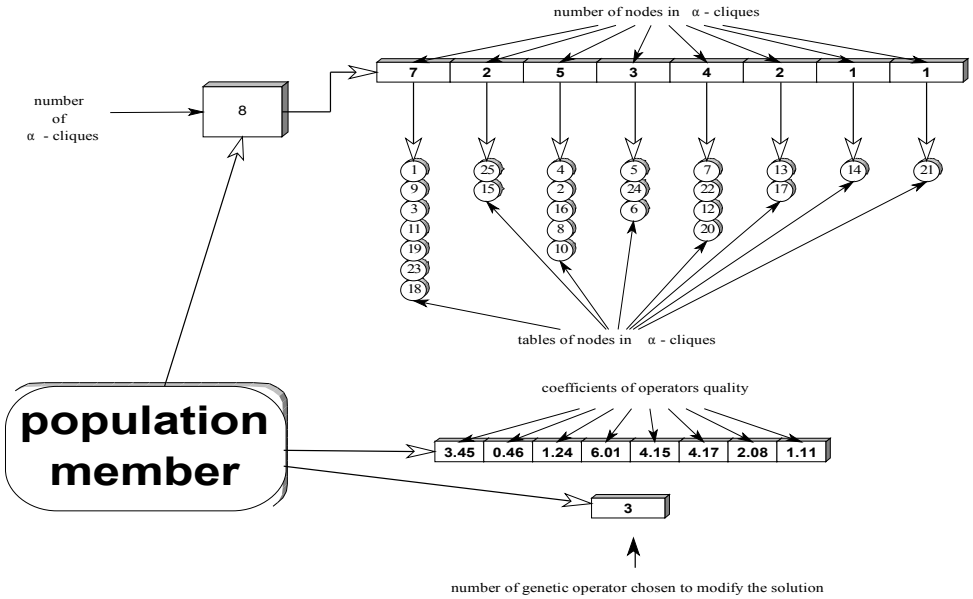

Figure 2. A structure of the population member.

Beside it, the member of the population contains several more data including: a vector of real numbers, which describe its knowledge about genetic operators and a number of the operator chosen for current iteration - more details abut them will be given later in this chapter.

3.2 Fitness Function

The problem's quality function is closely connected with the function, which evaluates the members of the population. In the problem several quality functions may be considered, depending on input data (binary or real) or what α-cliques one want to obtain (equal size or maximal size etc.). The fitness function does not have any punishment part for constraints violation, because forbidden solutions are not produced by initializing function or genetic operators. Thus all population members contain only α-cliques with their local values of α not less than the global value imposed on the solved problem.

$$max\, Q = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{l_i}\sum_{k=1}^{j} D[t_{ik}, t_{ij}] \tag{1}$$

where:
n – numbers of α-cliques in the valued solution;

l_i – number of nodes in the i-th α-clique;
D – the data array (incidence matrix);
t_{ij}, t_{ik} – nodes of the i-th α-clique.

Applying the fitness function (1) in computations gives as a result α-cliques of medium size (almost equal) and this version was used to solve the testing problem, while the fitness function (2) strongly promotes α-cliques of bigger size with some remaining very small ones:

$$max\, Q = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{l_i} \sum_{k=1}^{j} D[t_{ik}, t_{ij}] \tag{2}$$

where:
all symbols have the same meaning as in the formula (1).

3.3 Specialized Operators

The described data structure requires specialized genetic operators, which modify the population of solutions. Each operator is designed in such a manner that preserves the property of being α-clique for the modified parts of solutions. If modified solution violates the limitation of being α-clique, the operation is cancelled and no modification of solution is performed. This method makes it more difficult for the evolutionary algorithm to find satisfying solutions, due to possible bigger problems with local maximums, than standard method with penalty function, but it gives the certainty that computed solutions are always allowed.

Designed genetic operators are:

–mutation – an exchange of randomly chosen nodes in different α-cliques;
–movement of randomly chosen node to a different α-clique;
–"intelligent" movement – performed only if this operation gives better value of fitness function;
–concatenation – tries to concatenate (mainly small) α-cliques;
–also multiple versions of operators are applied (each operator is repeated several times on the selected individual during one epoch).

3.4 Evolutionary Algorithm Applied to Solve the Problem

Using of specialized genetic operators requires applying some method of sampling them in all iterations of the algorithm. In the used approach [7,13] it is assumed that an operator that generates good results should have bigger probability and more frequently effect the population. But it is very likely that the operator, that is good for one individual, gives worse effects for another, for instance because of its location in the domain of possible solutions. Thus every individual may have its own preferences. Every individual has a vector of floating point numbers, beside the encoded solution. Each number corresponds to one genetic operation. It is a measure of quality of the genetic operator. The higher the number is, the higher is the probability of the operator.

The ranking of qualities becomes a base to compute the probabilities of appearance and execution of genetic operators. This set of probabilities is also a base of experience of every individual and according to it, an operator is chosen in each epoch of the algorithm. Due to this experience one can maximize the survival chances of its offspring.

The applied selection method consists of two methods with different properties: a histogram selection (increases the diversity of the population) and a deterministic roulette (strongly promotes

best individuals) [13], which are selected in random during the execution of the algorithm. The probability of executing of the selection method is obtained from the formula (3).

$$p_{his}(t+1) = \begin{cases} p_{his}(t)*(1-a) & \text{for } R(t) > 3*\sigma(F(t)) \\ p_{his}(t)*(1-a)+0.5*a & \text{for } R(t) \geq 0.5*\sigma(F(t)) \wedge R(t) \leq 3*\sigma(F(t)) \\ p_{his}(t)*(1-a)+a & \text{for } R(t) < 0.5*\sigma(F(t)) \end{cases} \quad (3)$$
$$R(t) = \max(F_{av}(t) - F_{\min}(t), F_{\max}(t) - F_{av}(t))$$

where:

$p_{his}(t+1), p_{his}(t)$ - probability of histogram selection appearance in following iterations ($1-p_{his}(t)$ is a probability of the deterministic roulette method $p_{det}(t)$);

$F_{av}(t)$, $F_{min}(t)$, $F_{max}(t)$ - average, minimal and maximal values of fitness function in the population;

$\sigma(F(t))$ - standard deviation of fitness function ($F(t)$) in the population of solutions;

a - a small value to change probability $p_{his}(t)$, in simulations $a=0.05$.

If individuals in the population are described by a too small standard deviation of the fitness function ($\sigma(F(t))$) with respect to the extent of this function ($max(F_{av}(t)- F_{min}(t), F_{max}(t)- F_{av}(t))$), then it is desirable to increase the probability of appearance of the histogram selection. Conversely the probability of the deterministic roulette selection is increased. If parameters of the population are located in some range considered as profitable we may keep approximately the same probabilities of appearance for both methods of selection. It is important that always $p_{his}(t)+p_{det}(t)=1$- which means that some method of selection must be executed.

4. Experimental Results

The concept of α-clique and the described evolutionary algorithm have been used to solve a testing problem, described in this paragraph. The problem, we used as a testing example arose and was solved during airport designing, thus it is called an "airport example" [5], but it is an instance of a typical graph-clustering task.

Testing data is a symmetric matrix (27x27), which describes connections among 27 nodes (Figure 3) of graph representing designed airport. Graph nodes are some essential units of the airport, edges among them are flows of passengers, cargo, luggage etc.

The problem is to find some clusters (the number of clusters is not known in advance) of strongly connected units that can be placed closely with high-band connections. The idea of dividing a graph into α-cliques seems to be very promising for the described and similar tasks. In this approach α represents a percent of connections among nodes that must be covered in the extracted cluster and the algorithm tries to find clusters with the highest rates of “goods” exchange among nodes of this cluster, minimizing the total number of clusters.

This problem (also known as an instance of DSM - Design Structure Matrix [3,5]) has been also solved using other kinds of algorithms including greedy [5], 2-opt [5], 3-opt [5,14] or BEA – Bounded Energy Algorithm [5] or even EA [1,6,9,10,12,16]. The mentioned algorithms were used only as “preprocessing” methods. The preprocessing procedure helps to extract bounded groups of elements from the considered data, but some additional tool is required to separate the clusters [8,10].

Our approach, contrary to algorithms mentioned earlier in this chapter, delivers a complete method that produces final solutions with ready to use clusters. It uses a specialized EA not only as a preprocessing tool but also as a final clustering method. The evolutionary method

works quite fast and is rather simple, so it seems to be very promising and useful in solving such kinds of problems.

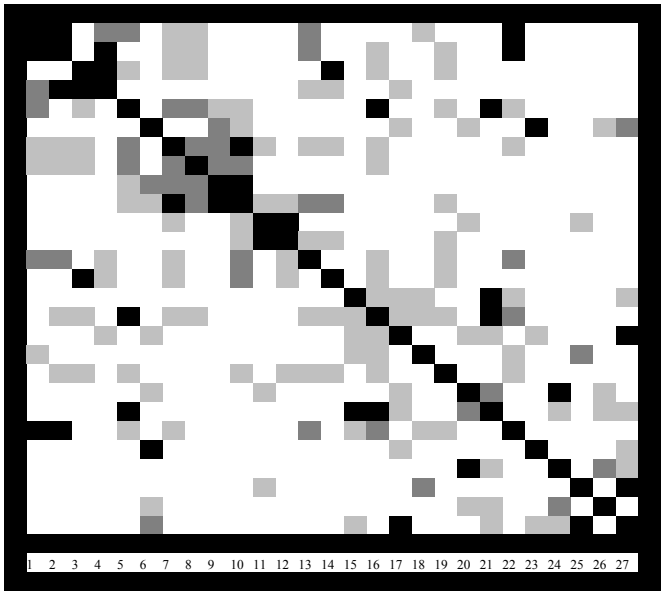

Figure 3. Original data.

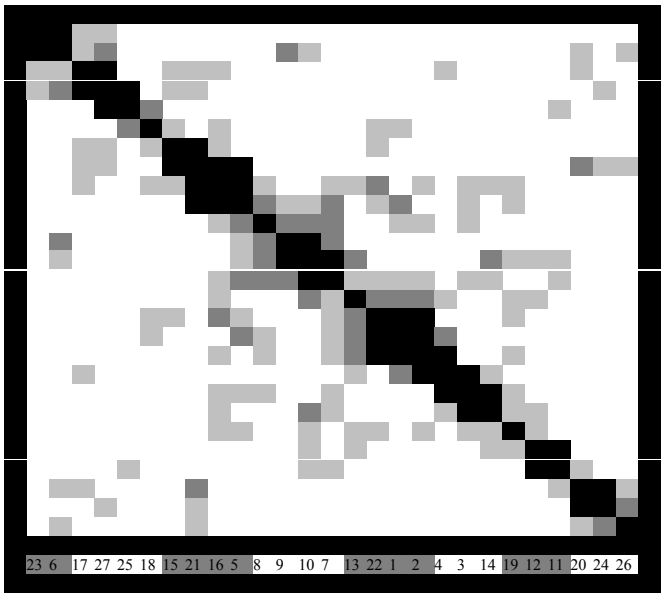

Figure 4. Ordering with EA (preprocessing) and extracting clusters with SCM [10].

Figures 4 and 5 show results obtained using initial data presented on Figure 3. Figure 4 presents results obtained using EA as a preprocessing tool and with extracting clusters using SCM [10] – simple clustering method. SCM is almost a manual method, it detects deep minimums in the histogram of data from preprocessed matrix and according to this generates clusters.

Figure 5 presents results generated by α-clique based method. The number of extracted clusters is equal to earlier method, but results are obtained easier and faster. Additional benefit of this method is that changing parameters of the fitness function (for instance using (1) or (2) or similar) or α we can easily influence the obtained solution (i.e. bigger or smaller clusters, almost equal or diverse).

On presented results numbers of columns marked with different colors in the last row of Figures 4 and 5 show extracted clusters of closely bounded elements of considered problem. Numbers of rows (and columns – symmetric problem) are the same as in Figure 3. Colors of small squares in the pictures denote the strength of connection between considered nodes (black-big strength of connection, white – no connection)

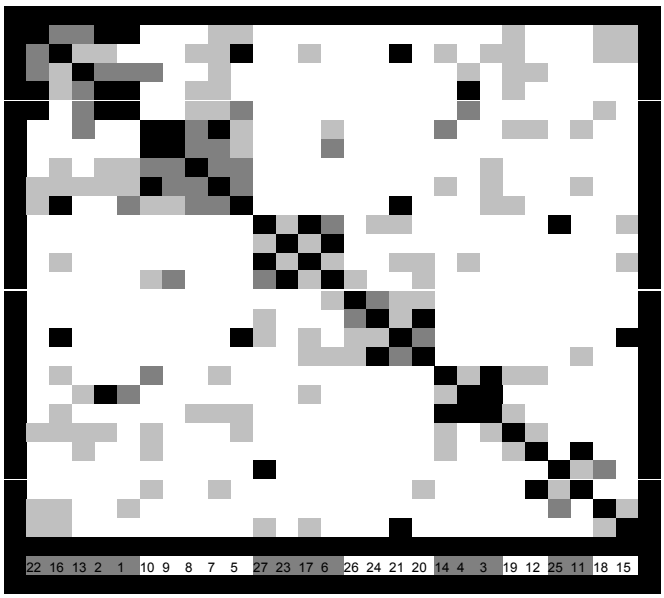

Figure 5. Clustering with the α-clique method..

5. Conclusions

The concept of α-clique gives new possibilities of separating "hardwired" structures from considered data, but determining α-cliques is a problem with large-scale complexity. Thus it seems to be justified to apply evolutionary algorithm to solve it. Experimental results confirm that applying α-cliques to determine concentrations of connections among objects delivers acceptable solutions and using specialized evolutionary algorithm makes it possible to obtain solution in reasonable time.

Bibliography

[1] Altus, S. S., Kroo, I. M., and Gage, P. J. A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems. *Journal of Mechanical Design*, Vol. 118, No. 4, pp. 486-489, 1996.

[2] Benson, S. J., and Ye, Y. Approximating maximum stable set and minimum graph coloring problems with the positive semidefinite relaxation. *Computational complexity: The problem of approximation*, Kluwer Academic Publishers, 2000.

[3] Browning, T. R. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management*, Vol. 48, No 3, 2001.

[4] Kloks, T., and Kratsch, D. Listing all minimal separators of a graph. *SIAM J. COMPUT.* Vol. 27, No. 3, June 1998, pp. 605-613, 1998.

[5] Lenkstra, J. K. Sequencing by enumerative methods. In *Mathematical Centre Tracts*, Amsterdam, 1977.

[6] McCulley, C., and Bloebaum, C. A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems. *Structural Optimization*, Vol. 12, No. 2-3, pp. 186-201, 1996.

[7] Mulawka, J., and Stańczak, J. Genetic Algorithms with Adaptive Probabilities of Operators Selection. In *Proceedings of ICCIMA'99*, New Delhi, India, pp. 464-468, 1999.

[8] Owsiński, J. W. On a new naturally indexed quick clustering method with a global objective function. In *Applied Stochastic Models and Data Analysis*, Vol. 6, pp 157-171, 1990.

[9] Potrzebowski, H., Stańczak, J., and Sęp, K. Evolutionary method in grouping of units with argument reduction. In *Proceedings of the 15th International Conference on Systems Science*, Vol. 3, Wrocław, Poland, pp. 29-36, 2004.

[10] Potrzebowski, H., Stańczak, J., and Sęp, K. Evolutionary methods in grouping of units. In *Proc. of the 4th International Conference on Recognition Systems CORES'05*, Springer-Verlag Berlin Heidelberg, pp. 279-286, 2005.

[11] Protasi, M. Reactive local search for the maximum clique problem. *Algorithmica*, vol. 29, no 4, pp. 610-637, 2001.

[12] Rogers, J L. Reducing Design Cycle Time and Cost Thorough Process Resequencing. *International Conference on Engineering Design ICED 1997*, Tampere, Finland, 1997.

[13] Stańczak, J. *Rozwój koncepcji i algorytmów dla samodoskonalących się systemów ewolucyjnych*, Ph.D. Dissertation, Politechnika Warszawska, 1999.

[14] Sysło, M. M., Deo, N. and Kowalik, J. S. *Algorithms of discrete optimization*, Prentice-Hall, 1983.

[15] Wilson, R. J. *Introduction to graph theory*. Addison Wesley Longman 1996.

[16] Yu, T. L., Goldberg, D. E., Yassine, A., and Yassine, C. A Genetic Algorithm Design Inspired by Organizational Theory. *Genetic and Evolutionary Computation Conference (GECCO) 2003*, Chicago, Illinois, USA, Publ. Springer-Verlag, Heidelberg, Lecture Notes in Computer Science, Vol. 2724/2003, pp. 1620-1621, 2003.