Barbara MAŻBIC-KULMA*, Henryk POTRZEBOWSKI*, Jarosław STAŃCZAK*, Krzysztof SĘP*

EVOLUTIONARY APPROACH TO FIND KERNEL AND SHELL STRUCTURE OF A CONNECTION GRAPH

Abstract: The theory of logistic transportation systems deals with models of phenomena connected with movement of goods and persons. The developed model of the transportation system is expected to simulate a real system, but also should help us to solve given transportation tasks. In order to describe transportation system (rail, bus or air), as a routine a connection graph would be used. Vertices of the graph can be train stations, bus stops etc.. The edges show direct connections between vertices. Its direct application can be difficult and computational problems can occur while one would try to organize or optimize such a transportation system. Therefore, a method of aggregation of such graph was introduced, using the general kernel and shell structure and its particular instances: hub-and-spoke and α-clique structured graphs of connections. These structures enable to concentrate and order the transport of goods/persons among vertices. To obtain these desired structures an evolutionary algorithm (EA) was applied. This method enables to reduce the number of analyzed vertices as well as arcs/edges of the graph.

Keywords: logistic network, kernel and shell graph, clique, evolutionary algorithm.

1. Introduction

The idea of kernel and shell structure of connection graph deals with the problem of separation of some highly bounded structures of a graph corresponding to same real logistic or transportation system defined in general by three essential components (Piasecki 1973, O'Kelly 1987, Coyle *et al.* 1994, Leszczyński 1994, Ambroziak 2000, Jacyna 2001, Kulma-Mażbic, Sęp 2005, Kulma-Mażbic *et al.* 2008):

- work task – necessity to relocate objects (cargo or /and persons),
- composition – type and number of elements describing the equipment and crew systems,
- organization – methods of system's elements reaction during task realization.

* Systems Research Insitute, Polish Academy of Sciences, Warsaw, Poland

The tasks of the freight-transport system are determined by the system customers' needs and they are described by the type and number of objects to freight, route for relocation of objects and the time of delivery.

There are many transport systems classifications, based, for instance, on:

- kind of transported objects (cargo, persons),
- quantity of transported objects,
- route of transported objects,
- transport means (railway, aircraft, vessel…).

The theory of transportation systems does not directly investigate physical phenomena connected with this domain, but its aim is to model and test models of transport systems. The model of transportation system should be accurate enough to replace the real system during the process of a solving particular problem. Mathematical description of transportation system is usually based on the notion of a connection graph. The vertices of this graph are railway stations, bus stations or airports, depending on the means of transport considered. Edges of this graph determine the presence of connections among vertices. As it can be easily noticed, a connection graph may have a big number of vertices and/or a big number of edges. The form of this graph has a big influence on transport organization. In this paper we propose the evolutionary method for optimization a logistic network introducing a *kernel and shell* structure, which is a generalization of well known *hub and spoke* structure and also similar approaches, including our α-clique structure (Protasi 2001, Potrzebowski *et al.* 2006b, 2007, 2008, Kulma-Mażbic *et al.* 2008).

The *kernel and shell* structure of a connection graph enables to concentrate flows of transported persons/goods among vertices. Figure 1 presents initial structure of connections before concentration. An adequate choice of several transit nodes and local connections could improve the transport system, reducing costs and increasing service efficiency. After the concentration process (Fig. 2), the graph of connections turned into a *kernel and shell* structure. The graph presented in Figure 1 may represent a structure of an existing traffic system, where the set of vertices corresponds to the set of traffic nodes and the set of edges correspond to the set of traffic connections. The *kernel and shell* structure presented in Figure 2 reduces the complexity of the management problem.

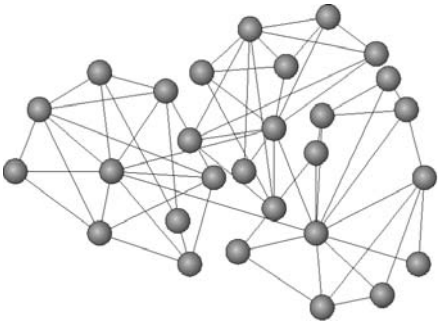

Fig. 1. *Input structure*

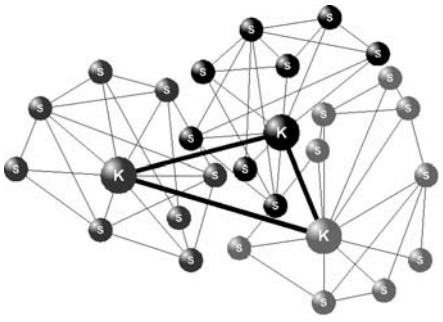

Fig. 2. *Relevant kernel and shell structure*

The advantages of such transport structure are:

- more frequent connections among points,
- lower average times of journeys,
- lower costs of transport,
- lower number of required transport means to assess all connections.

2. Graphs

Notions described below are based on Wilson (1996).

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 with $v_1 \neq v_2$ (Fig. 3).

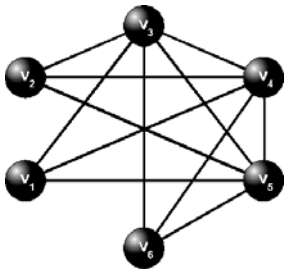

Fig. 3. *An example of a graph*

Two vertices in graph $G = (V, E)$ are called ***incident*** if for $v_i, v_j \in V$ there is $v_i, v_j \in E$. One vertex is ***incident*** to itself.

A ***sub-graph*** of graph $G = (V, E)$ is a graph $G' = (V', E')$, where $V' \subseteq V, V' \neq \emptyset$ and $E' \subseteq E$ such that for all $e \in E$ and $e = v_1, v_2$ if $v_1, v_2 \in V'$ then $e \in E'$.

A ***degree*** of vertex is the number of edges to which this vertex belongs.
Graph $G = (V, E)$ is a ***complete graph***, if for each pair of vertices there is an edge $e \in E$ between them.

A ***clique*** (*a complete sub-graph*) $Q = (V_q, E_q)$ in graph $G = (V, E)$ is a graph such that $V_q \subseteq V$ and $E_q \subseteq E$ and $Card(V_q) = 1$ or each pair of vertices $v_1, v_2 \in V_q$ fulfills the condition $v_1, v_2 \in E_q$ [4]. Each sub-graph of clique is a clique.

An α***-clique*** (Potrzebowski *et al.* 2006a).
Let $A = (V', E')$ be a sub-graph of graph $G = (V, E), V' \subseteq V, E' \subseteq E, k = Card(V')$ and let k_i be a number of vertices $v_j \in V'$ that $v_i, v_j \in E'$.

1) For $k = 1$ the sub-graph A of graph G is an α-***clique***(α).

2) For $k > 1$ the sub-graph A of graph G is an α-***clique***(α) if for all vertices $v_i \in V'$ fulfill the condition $\alpha \leqslant (k_i + 1)/k$, where $\alpha \in (0, 1]$.

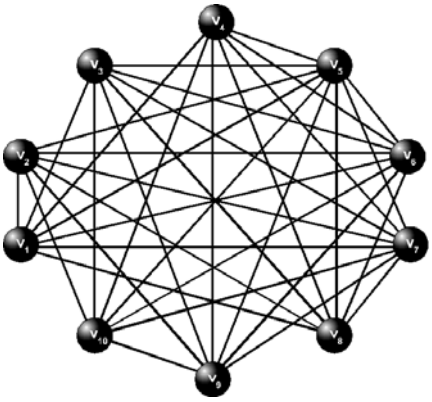

Fig. 4. *An example of $\alpha - clique(0.8)$*

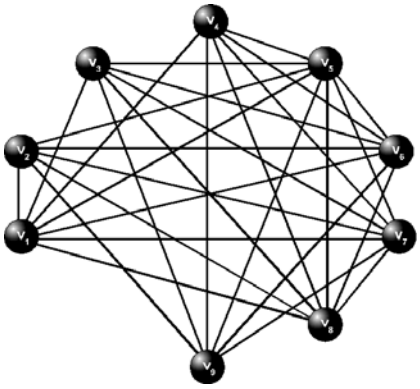

Fig. 5. *A sub-graph of graph from Figure 4 which is not $\alpha - clique(0.8)$*

Further we will use notion α-***clique*** in meaning α-***clique***(α) for earlier established α. As it can be seen in Figure 4 and Figure 5, a sub-graph of an α-$clique(0.8)$ is not an α-$clique(0.8)$, thus the property of being α-$clique(\alpha)$ may not be preserved by the sub-graphs of an α-$clique$. Let α-$clique$ $A = (V', E')$ be a graph with $\alpha > 0.5$, thus, for all vertices v_i belonging to α-$clique(\alpha)$ $k_i + 1 > 0.5k$.

The set theory implies the fact that for each pair of vertices, the sets of vertices incident with them have a non-empty intersection, so the α-$clique(\alpha)$ with $\alpha > 0.5$ constitutes a connected graph. If $\alpha = 0.5$, the obtained sub-graph may be disconnected.

A structure ***kernel and shell*** in graph $G(V, E)$ is composed of two graphs:

1) ***kernel*** – a sub-graph, which constitutes a group of strongly connected vertices $K(V_k, E_k)$, depending on actual needs it can be a clique, α-$clique$ or at last a connected sub-graph;

2) ***shell*** – a graph $S(V_s, E_s)$ where $V_s = V - V_k$ and $E_s = E - E_k$.

A ***hub-and-spoke*** structure (Fig 6c) is a graph $Hs = (G_h \cup G_s, E)$ where the subset G_h determines a (fully) connected graph (kernel) with the relevant subset of set E, each vertex of subset G_s has degree 1 and is connected exactly with one vertex from subset G_h (shell).

The hub and spoke is a particular case of a kernel and shell structure. This structure can be used in logistic models, where connections between nodes-"spokes" attached to its hub are not very important and direct connections are not necessary.

An α-***clique*** structure (Fig. 6b) of connection graph is also an instance of more general ***kernel and shell*** form. It consists of several peripheral (shell) α-$cliques$ G_α with desired values of α, connected with central (kernel) α-$clique$ G_c of strongly connected nodes with $\alpha \approx 1$.

The α-$clique$ structure should be considered when connections within selected sub-graph are also very important. The flow of goods among local nodes is too high to burden central nodes of G_c and also local connections must be assured. For logistic modeling we propose evolutionary methods that transform connection graph into an instance of the *shell and kernel* structure leading to the *hub-and-spoke* or α-$clique$ structures according to problem-specific restrictions.

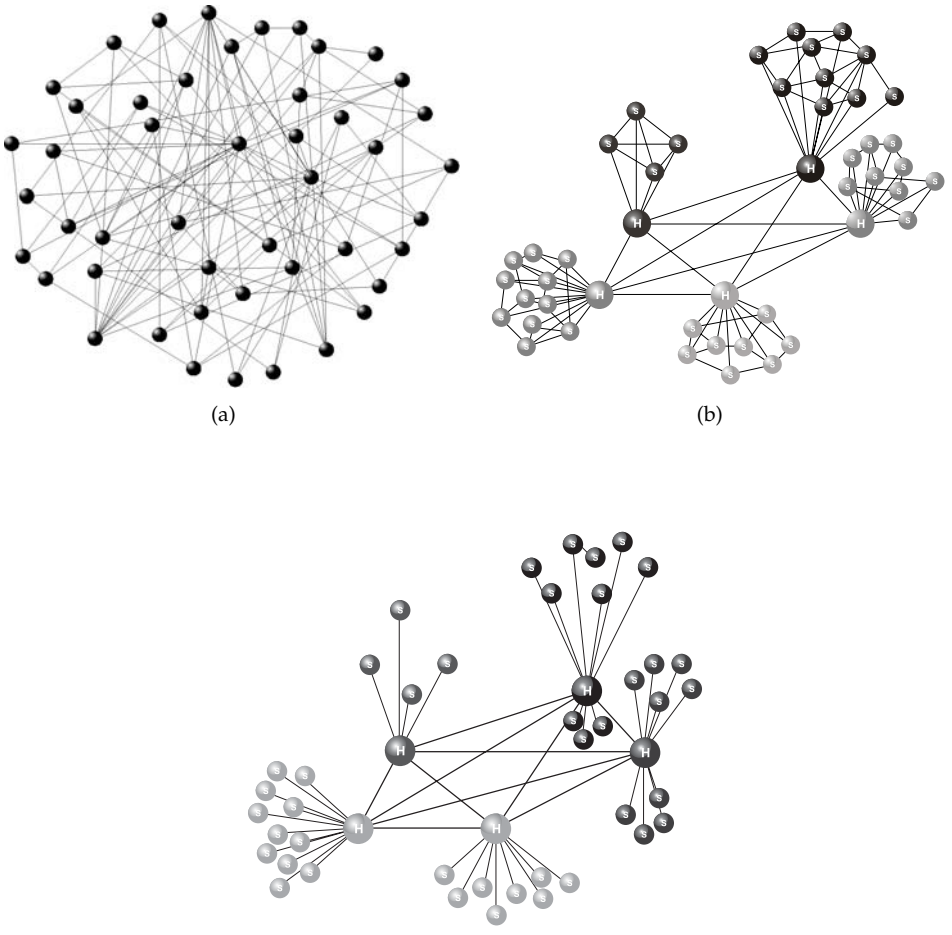

Fig. 6. *A source graph (a); the same graph organized to α-clique structure (b); the hub and spoke structure obtained from the source graph (c)*

3. The evolutionary methods to find desired connection graph structures

Standard evolutionary algorithm (EA) works in the manner shown in Algorithm 1, but this simple scheme requires many problem specific improvements to work efficiently. The adjustment of the genetic algorithm to the solved problem requires a proper encoding of solutions, an invention of specialized genetic operators for the problem, an accepted data structure and a fitness function to be optimized by the algorithm.

Algorithm 1

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

3.1. Individual representation

The problem encoding or in different words the individual representation depends on the desired graph structure to be obtained using EA. 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 population (Fig. 7) for the α-$clique$ structure contain their own solutions of the problem as a dynamic table of derived α-$cliques$ (their number may change during computations). Each element of this table (α-$clique$) has a list of nodes attached to this α-$clique$ and an element chosen as a hub for this α-$clique$. Each node is considered only once in one solution (population member), thus α-$cliques$ are separate.

The "hub and spoke" representation is encoded in similar way (Fig. 8) but "spokes" do not constitute α-$cliques$ but groups of nodes that are connected with their hubs. The sub-graph of "hubs" is an α-$clique$ with as big value of α as possible – ideally hubs should constitute a complete sub-graph, but in very difficult conditions, where connections between nodes are very sparse, it is admissible that the sub-graph of hubs constitutes simply a connected graph.

Besides a member of the population contains several more data items including: a vector of real numbers, which describe its knowledge about genetic operators and the number of the operator chosen to modify the solution in the current iteration. More details about genetic operators and the method of their evaluation will be given later in this paper.

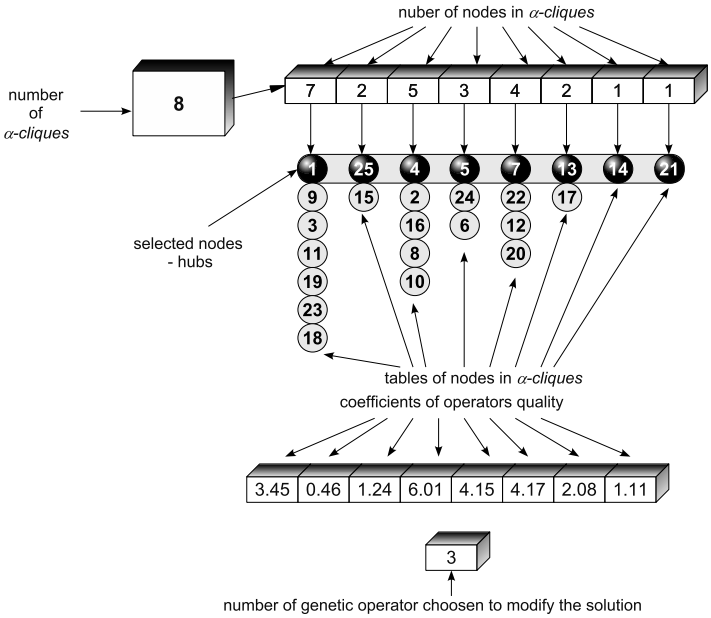

Fig. 7. *Structure of the population member for the α-clique structure*

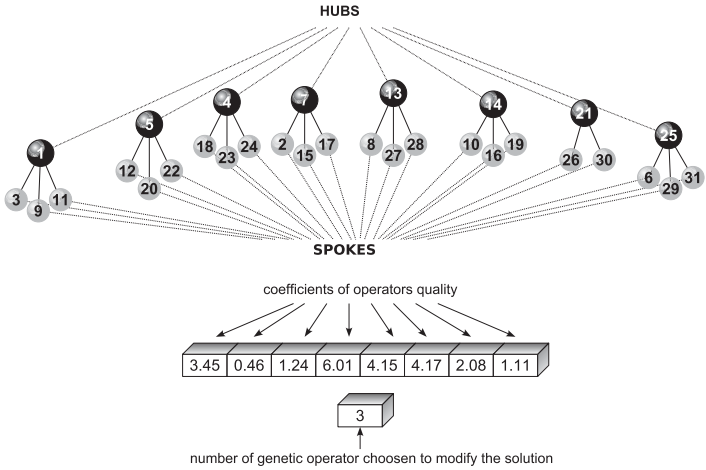

Fig. 8. *Structure of the population member for the hub and spoke structure*

3.2. Fitness functions

For both considered graph structures quality functions are a little artificial formulae, because solved problems are not pure optimization tasks and quality functions must precisely direct EA to find desired graph structures.

The problem specific quality function is closely connected with the fitness function, which evaluates the members of the population. Fitness function is a modified (scaled, moved, etc.) quality function, prepared for computations purposes in EA. In the first problem solved several quality functions may be considered, depending on input data (binary, integer or real) or what set of α-$cliques$ one wants to obtain (equal size or maximal size etc.).

The fitness function does not have to possess any punishment part for α-$clique$ constraint violation, because forbidden solutions are not produced during the process of population initialization or by the genetic operators. Thus all population members contain only valid α-$cliques$ with their local values of α not less than the global value imposed on the problem solved. For computer simulations we used the following quality function:

$$\max Q = \frac{1}{n}\sum_{i=1}^{n}\left(k_i - \left|\frac{k}{n} - k_i\right| + \frac{l_i}{k_i} - 1 + \frac{h_i}{n} - 1\right) \tag{1}$$

where:

n – number of α-$cliques$ in the solution evaluated,
k_i – number of nodes in the i^{th} α-$clique$,
k – number of nodes in the whole graph,
l_i – number of connections between the hub from i^{th} α-$clique$ and other nodes in this α-$clique$,
h_i – number of connections between hub i and other hubs.

The fitness function (1) promotes α-$cliques$ of size almost equal to average number of nodes in α-$cliques$, minimizing the number of obtained α-$cliques$, maximizing the number of connections among hubs and the number of connections between each selected hub and its α-$clique$. The second problem's quality function promotes solutions where a rather small sub-graph of hubs is (almost) fully connected and generated sets of spokes attached to their hubs have medium sizes:

$$\max Q = \frac{1}{n}\sum_{i=1}^{n}\left(k_i - \left|\frac{k-n}{n} - k_i\right| + \frac{h_i}{n} - 1\right) \tag{2}$$

where:

n – number of hubs in the solution evaluated,
k_i – number of nodes (spokes) attached to the i^{th} hub,
k – number of nodes in the whole graph,
h_i – number of connections between hub i and other hubs.

3.3. Specialized operators

The described data structures for both methods require specialized genetic operators, which modify the population of solutions.

For the first structure described, each operator is designed in a manner preserving the property of being α-$clique$ for the modified parts of solutions. If the modified solution violates the limitation of being α-$clique$, the operation is canceled and

no modification of solution is performed. With this method it is more difficult for the evolutionary algorithm to find satisfying solutions, due to the possible bigger problems with local maximums than method with penalty function, but it gives the certainty that computed solutions are always feasible.

The designed genetic operators are:

- mutation – an exchange of randomly chosen nodes in different α-$cliques$,
- movement of a randomly chosen node to a different α-$cliques$,
- "intelligent" movement – performed only if this modification gives a better value of the fitness function,
- concatenation – attempt to concatenate (mainly small) α-$cliques$,
- also multiple versions of operators are applied.

Additionally, each operator modifies elements selected as hubs for all α-$cliques$, using a simple mutation method.

The second described structure has similar operators, but different conditions are checked before they are performed When one node (spoke) is to be moved to another hub, first it must be checked if it has connection with this new hub. If no, the operation is canceled with consequences similar to previously described for α-$clique$ method.

The set of genetic operators in this case contains:

- mutation – an exchange of randomly chosen nodes in different sets of spokes,
- movement of a randomly chosen node to a different set of spokes,
- exchange of randomly selected hub for randomly selected spoke,
- concatenation – attempt to concatenate two sets of spokes,
- also multiple versions of operators are applied.

3.4. Evolutionary algorithm used to solve the problem

Use of specialized genetic operators requires having a method of selecting and executing them in all iterations of the algorithm. In the approach used (Potrzebowski *et al.* 2006a) 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 proper 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, besides the encoded solution. Each number corresponds to one genetic operation. It is a measure of quality of the genetic operator (a quality factor). The higher the factor, the higher the probability of using the operator. The ranking of quality factors becomes a basis for computing the probabilities of appearance and execution of genetic operators. Simple normalization of the vector of quality coefficients turns it into a vector of operator execution probabilities. This set of probabilities is also a basis of experience of every individual and according to it, an operator is chosen in each epoch of the algorithm. Due to the experience gathered one can maximize chances of its offspring to survive.

The method of computing quality factors is based on reinforcement learning (Cichosz 2000) (one of algorithms used in machine learning). An individual is treated as an agent, whose role is to select and call one of the evolutionary operators. When the selected i^{th} operator is applied, it can be regarded as an agent action a_i leading to a new state s_i, which, in this case, is a new solution. Agent (genetic operator) receives reward or penalty depending on the quality of the new state (solution).

The aim of the agent is to perform the actions, which give the highest long term discounted cumulative reward V^*:

$$V^{\Pi} = E_{\Pi} \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3}$$

$$V^* = \max_{\Pi} \left(V^{\Pi} \right) \tag{4}$$

The following formula can be derived from (3) and (4) and is used for evaluation purposes:

$$V\left(s_{t+1}\right) = V\left(s_t\right) + \alpha r_{t+1} + \gamma V^*\left(s_{t+1}\right) - V\left(s_t\right) \tag{5}$$

where:

Π – the strategy of the agent,
V^{Π} – discounted cumulative reward obtained using strategy Π,
E – expected value,
k – consecutive time steps,
t – current time,
$V(S_t)$ – a quality factor or discounted cumulative reward,
$V^*(S_{t+1})$ – estimated value of the best quality factor (in our experiments we take the value attained by the best operator),
α – a learning factor,
γ – a discount factor,
r_{t+1} – the reward for the best action, which is equal to the improvement of the quality of a solution after execution of the evolutionary operator.

In the presented experiments the values of α and γ were set to 0.1 and 0.2 respectively.

4. Obtained results of computer simulations

4.1. The testing data

Unfortunately, we did not have a real traffic data, thus we used a testing example from BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring) – Hiding Exact Solutions in Random Graphs (Bhoslib 2010). The chosen

problem was a graph with 450 vertices and 83 198 edges with the maximum clique size equal 30 (frb30-15-clq.tar.gz). The size of the problem is relatively big, but its complexity is similar to problems encountered during planning connections among bigger European cities.

4.2. Results obtained for the α-clique method

The first step to obtain the α-*clique* structure depends on the accepted value of parameter α. This step was made using the EA method and the results were as follows (Tab. 1):

Table 1. *Results of computations for various numbers of loops*

α		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0.75		3	150				138	139						2						
	num		3				2	1						3						
0.80		9	50				41	42	43	44	45	46	47	8						
	num		9				1	1	1	1	2	2	1	9						
0.85		15	24	29	30	34	22	25	26	27	28	31	32	13	14					
	num		1	2	10	2	1	2	2	7	1	1	1	8	7					
0.90		21	21	22			19	20	21					17	18	19	20			
	num		12	9			11	8	2					2	5	9	4			
0.95		29	14	15			15	14						23	24	25	26	27		
	num		15	16			15	14						3	4	4	11	7		
0.99		29	14	15			15	14						22	23	24	25	26	27	28
	num		15	16			15	14						1	1	5	4	10	5	3
1.00		29	14	15			15	14						24	25	26	27	28		
	num		15	16			15	14						5	12	4	6	2		

where:

column α – accepted value of α,
column 1 – number of obtained peripheral (shell) α-*cliques*,
column 2–5 – cardinality of particular α-*cliques* and number (num) of α-*cliques* with this cardinality,
column 6–12 – degree of kernel nodes in their α-*cliques* and number (num) of kernel nodes with this degree,
column 13–19 – number of kernel nodes connected with a particular kernel node and number (num) of such kernel nodes.

Using different values of α we obtained several different solutions. It is difficult to foresee a priori, taking in account only the value of α, which would be the best, but after conducted computations it is quite easy to choose the solution with the most acceptable parameters. The most important factor that influences the decision is the obtained number of kernel nodes. Probably for the problem of connecting bigger European cities the best solutions are the ones for $\alpha = 0.8$ or $\alpha = 0.85$ with 9 or 15

kernel nodes. Of course we can also try to find different numbers of kernel nodes accepting the values of α between the tested ones. The process of genetic computations lasts about 5–10 minutes, depending on the used operating system, machine and value of α, thus it is possible to compute solutions using several values of α, before accepting one.

Presented results provide a distinctly simplified structure of connections, for example in the case of $\alpha = 0.80$, we obtained output structure with 486 connections (edges) in place of the input structure with 83 198 connections. The lower number of connections implies faster and more frequent connections with lower costs for carriers and their clients.

5. Results obtained for the hub and spoke method

We used two methods to obtain this version of kernel and shell graph form. The first one was with numbers of hubs and spokes generated by the algorithm – it was the smallest number of hubs possible to cover the source graph. Its results are presented below.

Table 2. *Results obtained for algorithm with auto-partitioning of the source graph*

	1	**2**	**3**
	2	224	1
num		2	2

where:
column 1 – number of obtained sets of spokes,
column 2 – cardinality of particular sets of spokes and number (num) of sets of spokes with this cardinality,
column 3 – number of kernel nodes connected with a particular kernel node and number (num) of such kernel nodes.

It can be noticed that this version of algorithm gives to small possibilities of graph transformation. We obtain only one solution and in this case generated sets of spokes may be too big to be useful. But this results helps us to find the lower limitation of possible numbers of hubs. We can use it and set bigger numbers for the second method. The second method generates the kernel and shell structure with the given a priori number of hubs. Results obtained using this method are presented in Table 3.

Results collected in Table 3 show that the method with given number of hubs is much more flexible, because it is possible to obtain desired structure of transformed graph, while the first method gives only one solution. As it can be seen, for bigger numbers of hubs their sub-graph becomes not fully connected, but numbers of connections among hubs are very high (α of such α-$clique$ is close to 1). However, it is possible to obtain worse results for sparse graphs or for bigger number of hubs. It is necessary then to assess at least connectivity of that sub-graph.

Table 3. *Results obtained for desired numbers of hubs*

Number of hubs		1	2	3	4	5
4		4	112	111	3	
	num		2	2	4	
8		8	56	55	7	
	num		2	6	8	
16		16	28	27	15	
	num		2	14	16	
32		32	14	13	30	31
	num		2	30	12	20

where:
column 1 – number of obtained sets of spokes,
column 2–3 – cardinality of particular sets of spokes and number (num) of sets of spokes with this cardinality,
column 4–5 – number of kernel nodes connected with a particular kernel node and number (num) of such kernel nodes.

6. Conclusions

It is well known that for problems with large-scale complexity, there are no effective algorithms to solve them, specialized evolutionary methods are very efficient and give satisfying results. The results of the series of conducted experiments are rather optimistic, the parameter α introduced into the traditional notion of a clique gives rise a flexible tool that enables solving of the kernel and shell structure problem using α-cliques. Also traditional hub and spoke structure, which can be also treated as an instance of kernel and shell, can be easily obtained using evolutionary method. Presented methods can be very useful for developing logistic-transportation systems.

References

Ambroziak T. 2000. *O pewnych aspektach modelowania systemów transportowych* (On some aspects of modeling transportation systems). Prace Naukowe Transport, z. 44, OW PW, Warszawa.

Cichosz P. 2000. *Systemy uczące się* (Self-learning systems). WNT, Warszawa (in Polish).

Coyle J.J., Bardi E.J., Novack R.A. 1994. *Transportation (4th Edition)*. West Publishing Company, New York, pp. 402.

Hansen P., Mladenovic N., Urosevic D. 2004. *Variable neighborhood search for the maximum clique*. Discrete Applied Mathematics, Vol. 145, No. 1, pp. 117–125.

Jacyna M. 2001. *Modelowanie wielokryterialne w zastosowaniu do oceny systemów transportowych* (Multi-criteria modeling applied to transportation systems valuation). Prace Naukowe Transport, z. 47, OW PW, Warszawa.

O'Kelly M.E. 1987. *A quadratic integer program for the location of interacting hub facilities*. European Journal of Operational Research, 32, pp. 392–404.

Kulma-Mażbic B., Sęp K. 2005. *Problem wyboru węzłów tranzytowych w sieci lotniczej* (The problem of selection transit nodes in an airline network). Logistic Systems Theory and Practice, OW PW, pp. 341–348.

Kulma-Mażbic B., Potrzebowski H., Stańczak J., Sęp K. 2008. *Evolutionary approach to solve hub-and-spoke problem using α-cliques*. Evolutionary Computation and Global Optimization, Prace Naukowe PW, Warszawa, pp. 121–130.

Leszczyński J. 1994. *Modelowanie systemów i procesów transportowych* (Modelling of transportation precesses and systems). OW PW, Warszawa.

Piasecki S. 1973. *Optymalizacja systemów przewozowych* (Optimization of freight systems). WKiŁ, Warszawa.

Potrzebowski H., Stańczak J., Sęp K. 2007. *Separable decomposition of graph using alpha-cliques*. [in:] Kurzyński M., Puchała E., Woźniak M., Żołnierek A. (Eds), Computer recognition systems 2, Advances in soft computing, Springer-Verlag, Berlin - Heidelberg, pp. 386–393.

Potrzebowski H., Stańczak J., Sęp K. 2006. *Evolutionary Algorithm to Find Graph Covering Subsets Using α-Cliques, Evolutionary Computation and Global Optimization*. Prace Naukowe PW, Warszawa, pp. 351–358.

Potrzebowski H., Stańczak J., Sęp K. 2006. *Heurystyczne i ewolucyjne metody znajdowania pokrycia grafu, korzystające z pojęcia alfa-kliki i innych ograniczeń* (Heuristic and evolutionary methods of solving graph vertex cover problem using alpha-cliques and other constraints). [in:] Badania operacyjne i systemowe. Metody i techniki, Akademicka Oficyna Wydawnicza EXIT, Warszawa (in Polish).

Potrzebowski H., Stańczak J., Sęp K. 2008. *Evolutionary approach to solve hub-and-spoke problem using α-cliques*. Evolutionary Computation and Global Optimization, Prace Naukowe PW, Warszawa, pp. 121–130.

Protasi M. 2001. *Reactive local search for the maximum clique problem*. Algoritmica, Vol. 29, No. 4, pp. 610–637.

Stańczak J. 2003. *Biologically inspired methods for control of evolutionary algorithms*. Control and Cybernetics, Vol. 329, No. 2, pp. 411–433.

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

BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems. `http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm` (accessed November, 2009).