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

FINDING STRUCTURE KERNEL AND SHELL WITH PREDETERMINED CARDINALITY OF KERNEL SET, USING EVOLUTIONARY ALGORITHM

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 instance the α-clique structured graphs of connections. In the present approach, we use a predetermined number of communication hubs with the possibility of direct determining which nodes should become hubs or selecting them by the solving method. This structure allows to concentrate and order the transport of goods/persons among vertices and enables to reduce the number of analyzed vertices as well as arcs/edges of the graph. To obtain the desired structure, an evolutionary algorithm (EA) was applied.

Keywords: kernel and shell, clique, logistic network, 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 (O'Kelly 1987, Coyle *et al.* 1994, Leszczyński 1994, Ambroziak 2000, Jacyna 2001, Mażbic-Kulma, Sęp 2005, Mażbic-Kulma *et al.* 2008, Mażbic-Kulma *et al.* 2009):

- 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 (Potrzebowski *et al.* 2008, Potrzebowski *et al.* 2006a, 2007, 2008, Mażbic-Kulma, Sęp 2005).

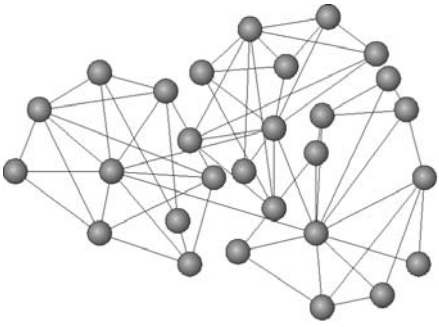


Fig. 1. *Input structure*

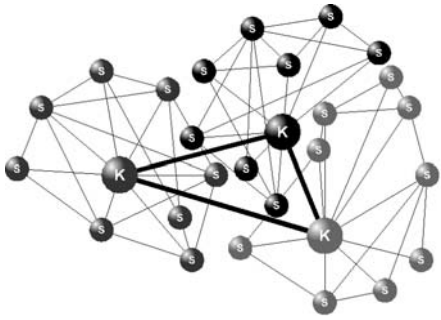


Fig. 2. *Relevant kernel and shell structure*

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 reduces the complexity of the management problem.

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.

In the present approach we use a predetermined number of communication hubs with the possibility of direct determining which nodes should become hubs or selecting them by the solving method. This approach can be more applicable in real-world situations, than described in previous papers method when the number of communication hubs is determined indirectly by imposed program parameters (mainly the parameter of α) [10].

The evolutionary algorithm is responsible for selecting the optimized configuration of shell nodes attached to their communication hubs and the best candidates for hubs, if they are not predefined by the user. The evolutionary method and obtained results are presented further in this paper.

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).

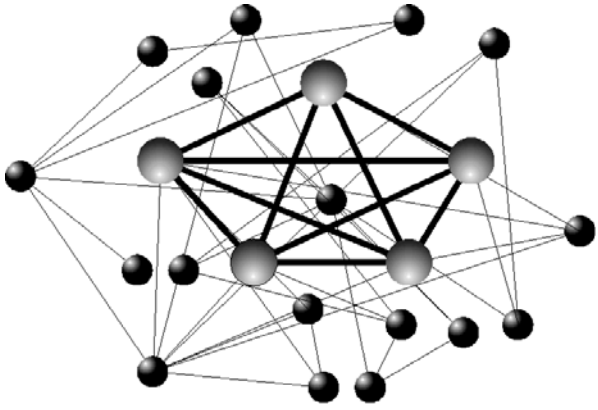


Fig. 3. *An example of a graph with marked α-clique*

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 $v_1, v_2 \in E_q$ (Hansen *et al.* 1994). Each sub-graph of clique is a clique.

An α***-clique*** (Potrzebowski *et al.* 2007).
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 $\alpha \in (0, 1]$.

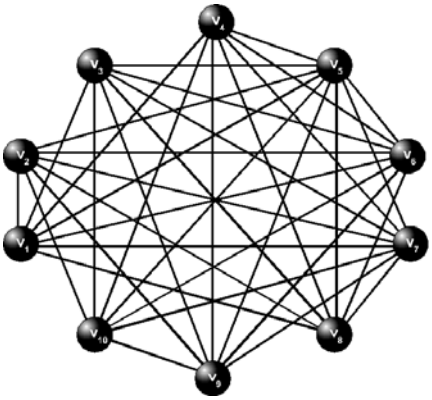


Fig. 4. *An example of α-clique(0.8)*

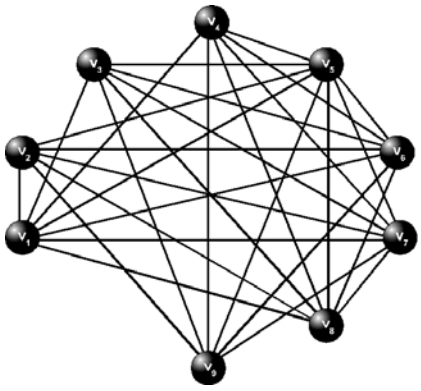


Fig. 5. *A sub-graph of graph from Figure 4 which is not α-dywiz clique(0.8)*

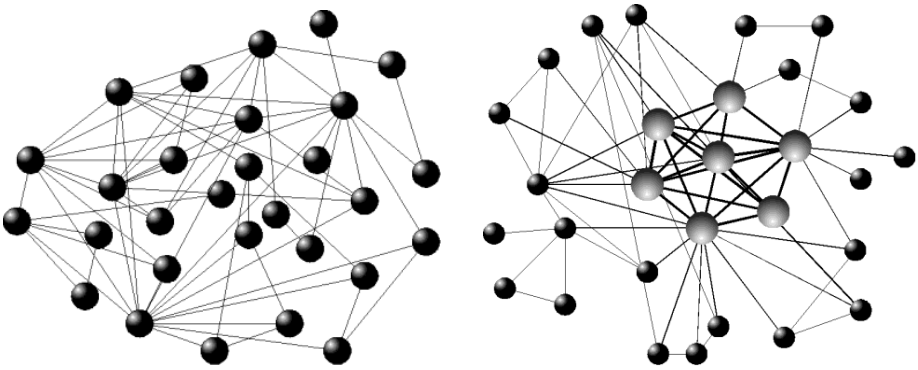
Further we will use notion α***-clique*** in meaning α***-clique***(α) for earlier established α. As it can be seen in Figures 4 and 5, a sub-graph of an α-$clique(0.8)$ is not an α-$clique(0.8)$, thus the property of being α-$clique(\alpha)$ may not be preserved by the sub-graphs of an α-$clique$. Let 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 $E_s = E - E_k$.

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 $\alpha \approx 1$.



(a) (b)

Fig. 6. *A source graph (a); the shell and kernel structure obtained from the source graph (b)*

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.

3. The evolutionary method to find the kernel and shell structure of connection graph

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, 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 of kernel and shell structure

The problem encoding or in different words the individual representation depends on the desired graph structure to be obtained using EA. In our approach the information about the problem is stored in an array of data that describes all connections among graph nodes. This array can be binary (an adjacency matrix 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. The "kernel and shell" representation of one solution is encoded in a manner presented in Figure 7.

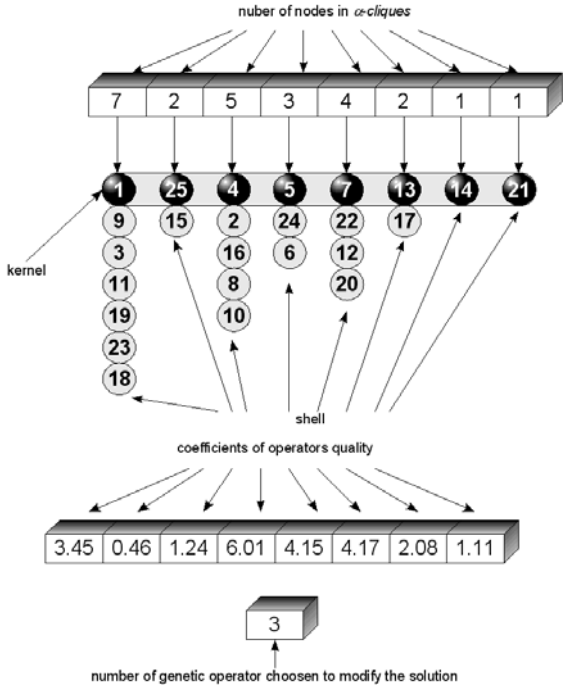


Fig. 7. *Structure of the population member*

The accepted data structure contains a table of predetermined number of α-*cliques* which constitute a shell of ordinary nodes. Each element of this table (α-*clique*) has a list of nodes attached to this α-*clique* and an element chosen as an

representative of this α-*clique* in the kernel structure – a communication hub. Each node is considered only once in one solution (population member), thus α-*cliques* are separate structures.

The kernel subgraph is an α-*clique* with as big value of α as possible – ideally hubs should constitute a complete subgraph (clique), but in very difficult conditions, where connections between nodes are very sparse, it is admissible that the kernel constitutes simply a connected graph. This condition is checked during computations and if is not satisfied, the penalty function significantly decreases a value of such solution. 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.

3.2. Fitness functions

The problem's quality function is responsible for obtaining the proper graph structure. The quality function is a little artificial formulae, because solved problem is not a pure optimization task and the quality function must precisely direct EA to find desired graph structure. Thus, many different formulae can be applied to obtain probably similar results.

In EA the fitness function is closely connected with problem's specific quality function. The fitness function evaluates the members of the population, it is a modified (scaled, moved, etc.) problem's quality function, prepared for computations purposes in EA. In the considered problem, several quality functions may be used, 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 has to possess a punishment part for the potential not connected kernel subgraph constraint violation. For computer simulations we used the following quality function:

$$
\begin{aligned}
Q_1 &= \alpha_{\min} - (1 - \alpha_K) - \frac{1}{k} \sum_{i=1}^{n} (k_i - l_i) \\
\max Q &= \begin{cases} \dfrac{Q_1}{m} & \text{if } Q_1 \geqslant 0 \\ Q_1 m & \text{if } Q_1 < 0 \end{cases}
\end{aligned}
\tag{1}
$$

where:

n – imposed number of α-cliques in the solution evaluated,
k – the number of nodes in the considered graph,
k_i – number of nodes in the ith α-clique,
l_i – number of connections between the hub from ith α-clique and other nodes in this α-clique,
m – number of connected subgraphs in kernel subgraph,
α_{min} – the minimum value of α in derived shell α-cliques,
α_K – the value of α in derived kernel α-clique.

The fitness function (1) promotes the kernel α-clique with the value of α_K as close to 1 as possible or if the transformed graph is very sparse, as a connected subgraph (with $m = 1$). The shell α-cliques should have their values of α as high as possible (maximization of α_{min} – the smallest value of α of all shell α-cliques) and as many as possible of their nodes should be connected with their communication hub.

3.3. Specialized operators

The described data structure requires specialized genetic operators, which modify the population of solutions.

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 communication hubs for all α-$cliques$, using a simple mutation method.

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.* 2007) 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^*:

$$V^{\Pi} = E_{\Pi} \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

$$V^* = \max_{\Pi} \left(V^{\Pi} \right) \tag{3}$$

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

$$V\left(s_{t+1}\right) = V\left(s_t\right) + \alpha r_{t+1} + \gamma V^*\left(s_{t+1}\right) - V\left(s_t\right) \tag{4}$$

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* 2009). The chosen problems were graphs with 450 vertices and 83 198 edges with the maximum clique size equal 30 (frb30-15-clq.tar.gz) and two complementary graphs the first with 4000 vertices and 572 774 with the maximum clique size equal 100 edges (frb100-40.mis.gz) and graph with 4000 vertices and 7 425 226 edges with Maximum Independent Set = 100 and Minimum Vertex Cover = 3900 (frb100-40.clq.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 kernel and shell method

In this approach the results depend on the accepted value of number of shell structures (α-cliques). The results were obtained using the EA method and the results were as follows (Tab. 1).

Table 1. *Comparison of obtained results*

the desired number of α-cliques		**min**	**max**	**mediana**	α_{min}	α_K**/conn**[1]
file name	vertices			edges		
frb30-15-mis.tar.gz	450			83 198		
5	Card of shell	53	149	77	0.79	1.00
	kernel's node degree in shell	51	142	74		
	kernel's node degree in kernel	5	5	5		
10	Card of shell	28	69	40	0.81	1.00
	kernel's node degree in shell	28	67	40		
	kernel's node degree in kernel	10	10	10		
20	Card of shell	16	29	22	0.86	0.95
	kernel's node degree in shell	16	29	22		
	kernel's node degree in kernel	19	20	19		
30	Card of shell	7	23	15	0.91	0.90
	kernel's node degree in shell	7	23	15		
	kernel's node degree in kernel	27	30	28		
35	Card of shell	1	22	15	1.00	0.89
	kernel's node degree in shell	1	22	15		
	kernel's node degree in kernel	31	34	33		
40	Card of shell	1	20	13	1.00	0.85
	kernel's node degree in shell	1	20	13		
	kernel's node degree in kernel	34	39	36		
file name	vertices			edges		
frb100-40.mis.gz	4000			572 774		
5	Card of shell	265	1818	520	0.03	1.00
	kernel's node degree in shell	54	197	76		
	kernel's node degree in kernel	5	5	5		
10	Card of shell	80	1561	181	0.03	0.40/+
	kernel's node degree in shell	31	189	45		
	kernel's node degree in kernel	4	6	4		
20	Card of shell	96	829	132	0.03	0.15/+
	kernel's node degree in shell	28	109	64		
	kernel's node degree in kernel	3	7	4		
30	Card of shell	59	394	130	0.04	
	kernel's node degree in shell	26	54	42		0.10/+
	kernel's node degree in kernel	3	8	5		
40	Card of shell	48	178	90	0.10	
	kernel's node degree in shell	23	85	47		0.10/+
	kernel's node degree in kernel	4	8	6		

Table 1. (cont.)

file name		vertices	edges			
frb100-40.clq.gz		4000	7 425 226			
5	Card of shell	224	1859	566	0.90	1.00
	kernel's node degree in shell	219	1800	547		
	kernel's node degree in kernel	5	5	5		
10	Card of shell	80	1586	178	0.90	1.00
	kernel's node degree in shell	80	1535	175		
	kernel's node degree in kernel	10	10	10		
20	Card of shell	60	1253	74	0.90	0.90
	kernel's node degree in shell	60	1213	74		
	kernel's node degree in kernel	18	20	19		
30	Card of shell	60	944	67	0.90	0.97
	kernel's node degree in shell	60	911	67		
	kernel's node degree in kernel	29	30	29		
40	Card of shell	55	668	64	0.90	0.93
	kernel's node degree in shell	55	652	64		
	kernel's node degree in kernel	37	40	39		

[1]The value of $\alpha > 0.5$ assures graph connectivity

5. 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 Politechniki Warszawskiej. Transport, z. 44, OW PW, Warszawa, pp. 29–56.

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

Cichosz P. 2000. *Systemy uczące się* [*Self-learning systems*]. WNT, Warszawa.

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

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 Politechniki Warszawskiej. Transport, z. 47, OW PW, Warszawa, pp. 3–139.

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.

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

Mażbic-Kulma 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 Politechniki Warszawskiej. Elektronika, z. 165, OW PW, Warszawa, pp. 121–130.

Mażbic-Kulma B., Potrzebowski H., Stańczak J., Sęp K. 2008. *Evolutionary approach to find kernel and shell structure of a connection graph*. Total Logistic Management, Vol. 2, AGH University of Science and Technology Press, Cracow, pp. 37–50.

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, Warszawa, pp. 341–348.

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.* [in:] *Evolutionary Computation and Global Optimization*, Prace Naukowe Politechniki Warszawskiej. Elektronika, OW 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.

Potrzebowski H., Stańczak J., Sęp K. 2008. *Evolutionary approach to solve hub-and-spoke problem using α-cliques*. [in:] *Evolutionary Computation and Global Optimization*, Prace Naukowe Politechniki Warszawskiej. Elektronika, OW 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.