TOTAL LOGISTIC MANAGEMENT No. 4 ● 2011 ● PP. 89–105

Barbara MAŻBIC-KULMA*, Jarosław STAŃCZAK*, Jan W. OWSIŃSKI*, Krzysztof SĘP*

EVOLUTIONARY APPROACH FOR OBTAINING THE HUB AND SPOKE STRUCTURE IN THE LOGISTIC NETWORK

Abstract: In order to describe transportation system, as a routine a connection graph would be used. Vertices of the graph can be train stations, bus stop, airports etc. The edges show direct connections between vertices. A direct application of such graph 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 a hub and spoke transformation method of the source graph. These structures allow to concentrate and order the transport of goods/persons among vertices and enable to reduce the number of analyzed vertices as well as edges of the graph. In the presented paper we continue our work on *kernel and shell* and its instance *hub and spoke* methods of connection graph transformation. In this paper we develop model of the transportation system using the *hub and spoke* method with predetermined, minimum and indirectly described numbers of hub nodes. To obtain the desired structures, several versions of specialized evolutionary algorithm (EA) were developed and applied.

Keywords: kernel and shell, hub and spoke, logistic network, evolutionary algorithm.

1. Introduction

The theory of logistic transportation systems deals with models of phenomena connected with movement of goods and persons. The idea of kernel and shell structure in graph of connections deals with the problem of separation some highly bounded structures in a graph corresponding to same real logistic system especially transportation network [1], [4], defined in general by three essential components [6], [7], [8], [9], [13], [14], [15], [16], [22]:

- 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 Institute Polish Academy of Science, 01-447 Warsaw, 6 Newelska Street, 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 [5], [10], [11], [12].

The theory of transportation systems does not directly investigate physical phenomena of this domain, but its goal is to build and to test models of transport systems [18]. The model of transportation system should be accurate enough to replace the real system during the process of solving a particular problem. Mathematical description of transportation system is usually represented by a graph of connections [19], [20], [21]. The vertices of this graph are railway stations, bus stations or airports, depending on considered means of transport. Edges of this graph are determined by a presence of connections among vertices. As it can be easily noticed, a connection graph may have a large number of vertices and edges. The form of this graph has a big influence on transport organization. In this paper we continue our work on the *kernel and shell* structure and its instance the *hub and spoke* ([8], [9]) structure using the evolutionary method for optimization a logistic network The *kernel and shell* structure is a generalization of well known *hub and spoke* structure and similar approaches, including an α-clique structure [14], [15], [19], [20], [21].

Using of the *kernel and shell* structure in a graph of connections enables to concentrate flows of transported persons or goods among vertices. The kernel subgraph constitutes a set of strongly connected vertices with cheap, fast or frequent connections (depending on the modeled transportation system), while the shell vertices are less frequently connected, mostly with their kernel vertices (hubs). The *kernel and shell* idea enables to eliminate many bilateral connections between the majority of vertices, instead of it only local connections between kernel vertices and local connections between kernel vertice and their shell vertices are required. Our approach deals with simple, undirected and unweighted graphs. These limitations are accepted to simplify the problem but presented methods can be extended and applied to that kinds of graphs.

Fig. 1 presents the initial structure of connections before transformation to the *kernel and shell* structure (concentration). An adequate choice of several transit nodes and local connections can improve the efficiency of transport system, reducing costs and increasing service quality. After the concentration process (Fig. 2), the graph of connections turned into a *kernel and shell* structure. The graph presented in Fig. 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,
- local connections are easier to synchronize, make timetables, etc.

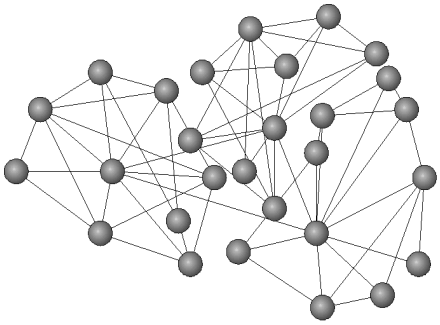


Fig. 1. *Input structure*

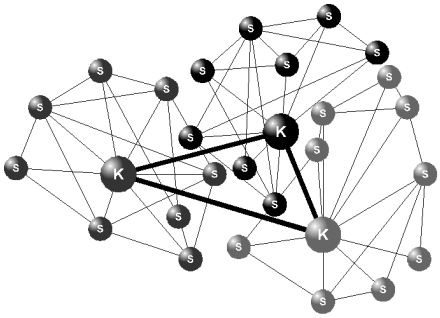


Fig. 2. *Relevant kernel and shell structure*

In this work we use a basic approach to transform the unstructured graph of connections into the hub and spoke structure.

The hub and spoke method of graph transformation is useful in situations where important are only connections between the kernel node (hub) and its shell nodes (spokes). All transfers within the shell are performed via the kernel node (hub). This method can also be used in several cases:

- predetermined number of communication hubs with the possibility of direct determining which nodes should become hubs or selecting them by the solving method;
- the minimum value of hubs which constitute at least a connected subgraph with all remaining nodes connected to their hubs (the number of hubs used in the previous point must be bigger than this minimum value);
- the number of communication hubs is determined indirectly by imposed program parameters, mainly the parameter of α (the hub subgraph must constitute an α-clique with imposed value of α).

The evolutionary algorithm has been chosen as a tool for graph transformations, because the transformation process is a hard computational problem and there are no efficient, dedicated algorithms of solving it. In our work the evolutionary algorithm (EA) 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 [24].

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 $v_1 \neq v_2$.

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 $v_1, v_2 \in E_q$ ([**?**]). Each sub-graph of clique is a clique.

An α***-clique*** [14], [19], [20], [21], [22].
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 $\alpha \in (0, 1]$.

It is simple to prove if $\alpha > 0.5$ then α-clique is a connected graph [23].
A structure ***kernel and shell*** in graph $G(V, E)$ is composed of two graphs:

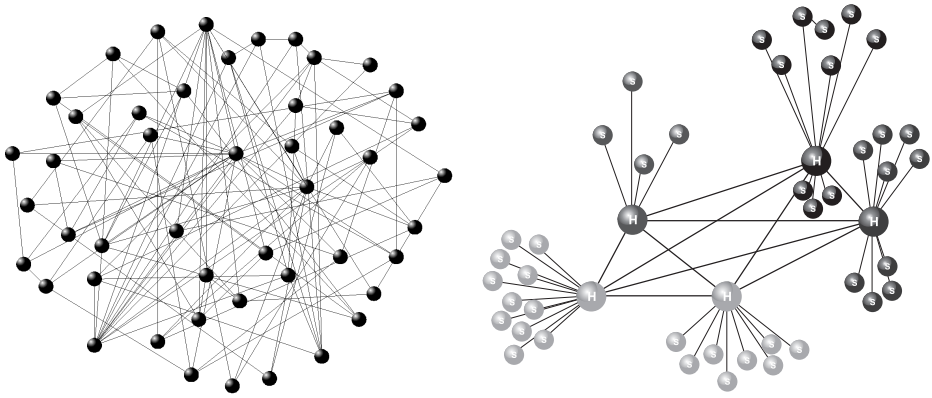
kernel – a subgraph, which constitutes a group of strongly connected vertices $K(V_k, E_k)$, depending on actual needs, possibilities or on the input graph structure, it can be a clique (ideally), an α-clique or at least a connected subgraph;

shell – a graph $S(Vs, Es)$ where $V_s = V - V_k$ and $E_s = E - E_k$, depending on optimized transportation system requirements, it can be an α-clique (including its kernel node) or a tree with the kernel node as root and shell nodes as leaves.

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* structure according to problem-specific restrictions.

A ***hub and spoke*** structure [9], [29] (Fig 3b) is a graph $H_s = (G_h \cup G_s, E)$ where the subset G_h determines at least a 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 direct connections between nodes-"spokes" attached to its hub are not very important and direct connections are not necessary. The hub and spoke structure can be derived using one of three possible methods. The first method uses a predetermined by some expert number of communication hubs with the possibility of direct determining which nodes should become hubs or selecting them by the solving method. The second method tries to find the minimum value of hubs which constitute at least a connected subgraph with all remaining nodes connected to their hubs. It must be noticed that the number of hubs used in the first method must be bigger than this minimum value.



(a) (b)

Fig. 3. *a source graph (a); the shell and kernel structure obtained from the source graph (b)*

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

The standard evolutionary algorithm works in the manner as shown in algorithm 2, but this simple scheme requires many problem specific improvements to work efficiently [17].

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

Algorithm 1: The standard evolutionary algorithm

```
Input:
        Input data
Output:
        Output data
begin
        Random initialization of the population of solutions.
        while stop condition is not satisfied do
            begin
                Reproduction and modification of solutions using genetic operators
                Valuation of obtained solutions
                Selection of individuals for the next generation
            end;
end;
```

Evolutionary algorithms are often used to solve difficult graph problems such as graph coloring, TSP, graph partitioning, maximum clique searching, etc. [2], [24] thus it seems fully justified to use the evolutionary algorithm in graph clustering problem.

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 a connection graph is a general structure, several particular instances of such structure are described in this work. Each case of these structures requires specific encoding method. Generally, described encoding methods are rather similar but it is necessary to point out the differences.

In the 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 computational purposes in the EA. The problem's quality function is responsible for obtaining the proper graph structure. The quality functions in considered problems are little artificial formulae, because solved problems are not pure optimization tasks, but transformation graph structure to the desired form can be treated as an optimization problem. The quality function must precisely direct EA to find desired graph structure. Thus, several quality functions may be used, depending on input data (binary, integer or real) or what set of kernel nodes or shell nodes one wants to obtain (equal size or maximal size, the minimum number of kernel nodes, etc.). Usually the fitness function has to contain a punishment part for the potential not valid solutions.

Another important problem is designing genetic operators for accepted data structure, taking into account constraints imposed on solutions. It is obvious that standard crossover and mutation are not proper, so problem-specific, specialized operators must be prepared to efficiently solve described problems. This section describes applied methods of solutions' modification in detail.

3.1. The *hub and spoke* structure

As it has been noticed, the *hub and spoke* structure of a connection graph is useful when direct transfers among spokes in one shell subgraph are not necessary. Thus, shell subgraphs need not constitute α-cliques but a structure of a root and attached leaves or in different words a star or a hub and spoke structure.

3.1.1. The *hub and spoke* structure with predetermined number of kernel nodes

The *hub and spoke* representation is presented in Fig. 4. The *spokes* do not constitute α-cliques but groups of nodes that are connected with their hubs. The subgraph of *hubs* is an α-clique with as big value of α as possible – ideally hubs should constitute a complete subgraph, but in very difficult conditions, where connections between nodes are very sparse or are determined as existing junction nodes (for instance railway stations), it is admissible that the subgraph of hubs constitutes simply a connected

graph. The kernel nodes can be explicitly assigned or only the number of them may be imposed.

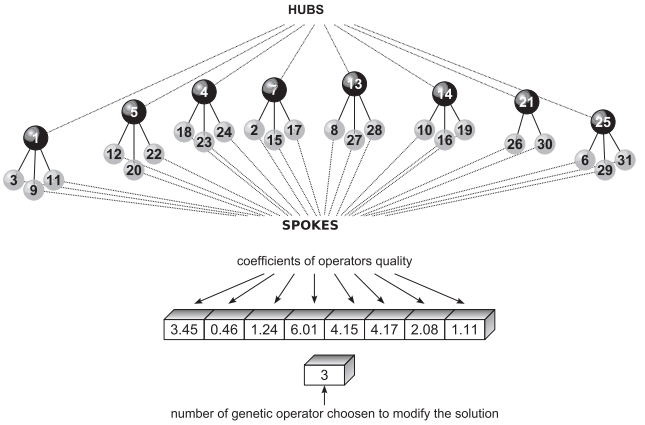


Fig. 4. *Structure of the population member for the hub and spoke structure with predetermined number of kernel nodes*

A member of the population as well, 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.

The *hub and spoke* structure with predetermined number of kernel nodes problem's quality function promotes solutions where a rather small subgraph of hubs is (almost) fully connected and generated sets of spokes attached to their hubs have medium sizes:

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

where:

n – predetermined (constant) number of hubs in the solution evaluated,
m – number of connected subgraphs in kernel subgraph,
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.

The fitness function (1) promotes bigger shell subgraphs (spokes), ideally of size almost equal to average number of spokes ($|(k-n)/n - k_i|$), assuring connectivity of kernel subgaph ($1/m$ – for connected subgraph m should be 1), maximizing the number of connections among hubs ($h_i/n - 1$).

This problem can be solved using similar operators to the α-clique methods, but different conditions are checked before they are performed. When one node (spoke) is to be moved to another hub's shell, first it must be checked if it has connection with this new hub. If not, the operation is canceled and no modification of solution is performed.

In this case the set of genetic operators in this case contains:

1) mutation – an exchange of randomly chosen nodes in different sets of spokes;
2) movement of a randomly chosen node to a different set of spokes;
3) exchange of randomly selected hub for randomly selected spoke – this operator is inactive when kernel nodes are explicitly assigned;
4) also multiple versions of operators are applied.

The problem arises when the predetermined number of kernel nodes is lower than the minimal value (see p. 3.1.3), that assures that all shell nodes will be attached to their kernel hubs. This problem can be solved in two ways. The first allows that the final result contains unattached nodes. The second increases the number of kernel nodes to obtain connected graph of connections. These methods are realized using modified forms of quality function (1) with the punishment part for unattached shell nodes or additional kernel nodes.

3.1.2. The *hub and spoke* structure with indirectly imposed number of kernel nodes

The case of *hub and spoke* structure with indirectly imposed size of kernel subgraph is encoded in manner presented on Fig. 5. The number of kernel nodes (hubs) is unknown in advance and varying during computations. The algorithm must find the best value and kernel candidates optimizing quality function (2). The data structure contains a dynamic table of chosen kernel nodes (communication hubs) and lists of *spokes* which constitute shells of ordinary nodes. As in the previously considered cases, each node is considered only once in one solution.

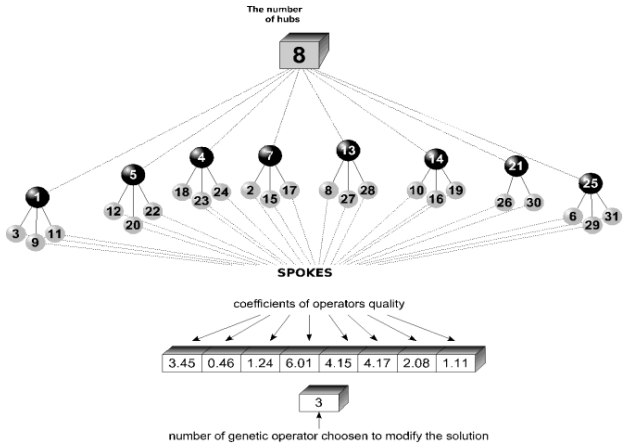


Fig. 5. *Structure of the population member for the hub and spoke structure with indirectly imposed number of kernel nodes*

For computer simulations we used the following quality function:

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

where:
n – number of hubs in the solution evaluated,
m – number of connected subgraphs in kernel subgraph,
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.

The fitness function (2) promotes bigger shell subgraphs (spokes), ideally of size almost equal to average number of spokes ($|(k-n)/n - k_i|$), assuring connectivity of kernel subgraph ($1/m$ – for connected subgraph m should be 1), maximizing the number of connections among hubs ($h_i/n - 1$).

The set of genetic operators in this case contains:

1) mutation – an exchange of randomly chosen nodes in different sets of spokes;
2) movement of a randomly chosen node to a different set of spokes (random and "intelligent" version, "intelligent" version performs changes in the individual only if new set of spokes are better connected with their kernels than before this operation);
3) exchange of randomly selected hub for randomly selected spoke (as in the previous case random and "intelligent" version);
4) concatenation – attempt to concatenate two sets of spokes (tries to minimize the number of the kernel nodes);
5) also multiple versions of operators are applied.

3.1.3. The *hub and spoke* structure with the minimum number of kernel nodes

The *hub and spoke* structure with the minimum size of kernel subgraph is a special case of the structure with indirectly imposed number of kernel nodes, but this problem is computationally more difficult to solve. The problem encoding is identical as in the case with the indirectly determined size of kernel subgraph (Fig. 5) but the optimized fitness function (3) is different:

$$\min Q = n \cdot m \tag{3}$$

where:
n – number of hubs in the solution evaluated,
m – number of connected subgraphs in kernel subgraph.

The fitness function (3) promotes the smallest set of connected kernel nodes with all spokes attached to their hubs.

The set of genetic operators in this case contains:

1) mutation – an exchange of randomly chosen nodes in different sets of spokes;

2) movement of a randomly chosen node to a different set of spokes (random and "intelligent" version, "intelligent" version performs changes in the individual only if new set of spokes are better connected with their kernels than before this operation);
3) exchange of randomly selected hub for randomly selected spoke (as in the previous case random and "intelligent" version);
4) concatenation – attempt to concatenate two sets of spokes (tries to minimize the number of the kernel nodes);
5) also multiple versions of operators are applied.

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 [23] 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 [3] (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^*:

$$V^{\Pi} = E_{\Pi} \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4}$$

$$V^* = \max_{\Pi} \left(V^{\Pi} \right) \tag{5}$$

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

$$V(s_{t+1}) = V(s_t) + \alpha r_{t+1} + \gamma V^*(s_{t+1}) - V(s_t) \tag{6}$$

where:

Π – represents the strategy of the agent,
V^{Π} – represents discounted cumulative reward obtained using strategy Π,
E – represents expected value,
k – represents consecutive time steps,
t – represents current time,
$V(S_t)$ – is 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 – is a learning factor,
γ – is a discount factor,
r_{t+1} – represents 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.

5. Obtained results of computer simulations

5.1. 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 [30]. 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 considered problems are relatively big, but its complexity is similar to problems encountered during planning connections among bigger European cities for instance.

5.2. Results obtained for the hub and spoke method

5.2.1. The problem with the minimum number of kernel nodes

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 1. *Results obtained using algorithm searching the minimum partitioning of the source graph*

Obtained number of kernel nodes		**min**	**max**	**mediana**
Graph problem name	Vertices			Edges
frb30-15-1	450			83 198
	Card of shell	2	2	2
2	kernel's node degree in shell	67	381	2
	kernel's node degree in kerne	2	2	2
Graph problem name	Vertices			Edges
frb100-40_mis	4 000			572 774
	Card of shell	4	291	60
45	kernel's node degree in shell	4	291	60
	kernel's node degree in kerne	2	7	4
Graph problem name	Vertices			Edges
frb100-40	4 000			7 425 226
	Card of shell	2	2	2
2	kernel's node degree in shell	2 069	2 929	2
	kernel's node degree in kerne	2	2	2

It can be noticed that this version of algorithm gives too small possibilities of graph transformation. We obtain only one solution for each example 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.

5.2.2. The problem with imposed number of kernel nodes

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 Tab. 2. Due to input graph structure, some hub and spoke structures with predefined numbers of kernel nodes may not be achieved without leaving unattached shell nodes (and required number of kernel nodes) or without increasing number of kernel nodes (without unattached nodes), for instance it would happen if we wanted a hub and spoke structure with 30 kernel nodes for the frb100-40-mis problem. In presented results of we considered results obtained in the previous section and requested kernel values are bigger than obtained minimal values.

Table 2. *Results obtained for desired numbers of hubs*

The desired number of kernel nodes		**min**	**max**	**mediana**	α_K/**conn**[1]
	Graph problem name frb30-15-1	Vertices 450		Edges 83198	
5	Card of shell kernel's node degree in shell kernel's node degree in kernel	89 89 5	89 89 5	89 89 5	1.00
10	Card of shell kernel's node degree in shell kernel's node degree in kernel	44 44 10	44 44 10	44 44 10	1.00
20	Card of shell kernel's node degree in shell kernel's node degree in kernel	21 21 20	22 22 20	21 21 20	1.00
40	Card of shell kernel's node degree in shell kernel's node degree in kernel	10 10 37	11 11 40	10 10 39	0.93
50	Card of shell kernel's node degree in shell kernel's node degree in kernel	8 8 46	8 8 50	8 8 48	0.92
100	Card of shell kernel's node degree in shell kernel's node degree in kernel	3 3 89	4 4 97	3 3 92	0.89
	Graph problem name frb100-40-mis	Vertices 4000		Edges 572 774	
50	Card of shell kernel's node degree in shell kernel's node degree in kernel	62 62 2	113 113 11	79 79 5	0.04/+
60	Card of shell kernel's node degree in shell kernel's node degree in kernel	65 65 2	66 66 13	66 65 5	0.03/+
75	Card of shell kernel's node degree in shell kernel's node degree in kernel	26 26 3	53 53 15	39 39 8	0.03/+
100	Card of shell kernel's node degree in shell kernel's node degree in kernel	18 18 2	48 48 8	39 39 15	0.02/+
200	Card of shell kernel's node degree in shell kernel's node degree in kernel	13 13 5	27 27 29	19 19 16	0.02/+

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

Table 2. *Results obtained for desired numbers of hubs – cont'd*

The desired number of kernel nodes		**min**	**max**	**mediana**	α_K**/conn**
Graph problem name		Vertices		Edges	
frb100-40		4000		7 425 226	
5	Card of shell	799	799	799	1.00
	kernel's node degree in shell	799	799	799	
	kernel's node degree in kernel	5	5	5	
10	Card of shell	399	399	399	1.00
	kernel's node degree in shell	399	399	399	
	kernel's node degree in kernel	10	10	10	
20	Card of shell	199	199	199	1.00
	kernel's node degree in shell	199	199	199	
	kernel's node degree in kernel	20	20	20	
50	Card of shell	79	79	79	1.00
	kernel's node degree in shell	79	79	79	
	kernel's node degree in kernel	50	50	50	
100	Card of shell	39	39	39	0.99
	kernel's node degree in shell	39	39	39	
	kernel's node degree in kernel	99	100	99	
200	Card of shell	19	19	19	0.97
	kernel's node degree in shell	19	19	19	
	kernel's node degree in kernel	194	198	196	

Results collected in Tab. 2 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 for each case. As it can be seen, for bigger numbers of hubs their subgraph 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 subgraph.

5.2.3. The problem with indirectly imposed number of kernel nodes

Results obtained in this case are similar to those obtained in the case of the minimum kernel size, different results are only for the sparse graph frb100-40-mis, where additional limitations (connectedness of kernel subgraph, requirement that all spokes must be connected with their hubs with proper structure of shell subgraphs) enlarged the number of obtained kernel nodes.

Table 3. *Results obtained for algorithm with auto-partitioning of the source graph*

Obtained number of kernel nodes		**min**	**max**	**mediana**	α_K**/conn**[2]
	Graph problem name	Vertices		Edges	
	frb30-15-1	450		83 198	
	Card of shell	224	224	224	
2	kernel's node degree in shell	224	224	224	1.00
	kernel's node degree in kernel	2	2	2	
	Graph problem name	Vertices		Edges	
	frb100-40-mis	4000		572 774	
	Card of shell	71	89	79	
50	kernel's node degree in shell	71	89	79	0.04/+
	kernel's node degree in kernel	2	20	8	
	Graph problem name	Vertices		Edges	
	frb100-40	4000		7 425 226	
	Card of shell	1999	1999	1999	
2	kernel's node degree in shell	1999	1999	1999	1.00
	kernel's node degree in kernel	2	2	2	

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

6. Conclusions

The results of the series of conducted experiments show that hard computational problems, like transformation a logistic network into the *kernel and shell* structure can be easily done using evolutionary algorithms. Also the *hub and spoke* structure, which can be treated as an instance of kernel and shell, can be easily obtained using the evolutionary method.

Presented methods can be very useful for developing logistic and transportation systems.

References

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

[2] Chen, Z.-Q., Wang, R.-L. and Okazaki, K., (2008) *An Efficient Genetic Algorithm Based Approach for the Minimum Graph Bisection Problem*, IJCSNS International Journal of Computer Science and Network Security, Vol. 8 No.6, pp 118–124

[3] Cichosz P., (2000) *Systemy uczące się* (in Polish), (eng.: Self-learning systems) WNT, Warszawa

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

[5] Fechner I., *Najlepsze praktyki w logistyce*, LOGISTICS 2006

[6] Horner, M. and M.E. O'Kelly, (2001) *Embedding economy of scale concepts for hub network design*. Journal of Transport Geography, 9 (4), pp. 255–265

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

[8] 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

[9] O'Kelly M.E. and Bryan D., (2002) *Interfacility interaction in models of hubs and spoke networks*. Journal of Regional Science, 42 (1), pp. 145–165

[10] Krawczyk S., *Koordynacja procesów w sieciach logistycznych*, Wybrane zagadnienia logistyki stosowanej Kraków 2006

[11] Mażbic-Kulma B., Piasecki S., *Geneza i organizacja sieci centrów logistycznych*, Wybrane zagadnienia logistyki stosowanej Kraków 2004

[12] Mażbic-Kulma B., Pogorzelec A., Komorowska E., (2005) *Lokalizacja obiektów. Wybrane modele, algorytmy i zastosowania*, Badania Systemowe IBS PAN tom 39

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

[14] 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 PW, Warszawa

[15] Mażbic- Kulma B, Potrzebowski H., Stańczak J., Sęp K., (2009) *Evolutionary approach to find kernel and shell structure of a connection graph*, Total Logistic Management AGH, Cracow, pp. 37–50

[16] McCulley C., Bloebaum C. A., (1996) *Genetic tool for optimal design sequencing in complex engineering systems*, Structural Optimization, Vol. 12, No. 2–3

[17] Min H. and Gou Z., (2004) *The location of hub-seaports in the global supply chain network using a cooperative competition strategy*. Integrated supply management, Vol 1 No.1, pp. 51–63

[18] Michalewicz Z., (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, Berlin Heidelberg

[19] Owsiński J.W., (1990) *On a new naturally indexed quick clustering method with a global objective function*, Applied Stochastic Models and Data Analysis,Vol. 6

[20] Piasecki S. (1973) *Optymalizacja systemów przewozowych*, (eng.: Optymization of freight systems), WKiŁ, Warszawa

[21] Potrzebowski H., Stańczak J., Sęp K., (2005) *Evolutionary method in grouping of units*, Proceedings of the 4th International Conference on Recognition Systems CORES'05, Springer-Verlag, Berlin-Heidelberg.

[22] 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

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

[24] 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*, in: Advances in soft computing Springer-Verlag, Berlin-Heidelberg, pp. 386–393

[25] 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.

[26] Stańczak J., (2003) *Biologically inspired methods for control of evolutionary algorithms*, Control and Cybernetics, 32(2), pp. 411–433

[27] Talbi E.-G. and Bessiere P., (1991) *A parallel genetic algorithm for the graph partitioning problem*, Proceeedings of the 5th International conference on Supercomputing, ACM, New York, pp. 312–320

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

[29] `http://people.hofstra.edu/geotrans/eng/ch3en/conc3en/hubspokederegulation.html`

[30] `http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm`