O P E R A T I O N S R E S E A R C H A N D D E C I S I O N S
No. 2 2013
DOI: 10.5277/ord130208

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

THE KERNEL AND SHELL STRUCTURE AS A TOOL FOR IMPROVING THE GRAPH OF TRANSPORTATION CONNECTIONS

A model of a transportation system is expected to be useful in simulations of a real system to solve given transportation tasks. A connection graph is routinely used to describe a transportation system. Vertices can be train stations, bus stops, airports etc. The edges show direct connections between vertices. A direct approach can be difficult and computational problems can arise in attempts to organize or optimize such a transportation system. Therefore, a method for aggregating such graphs was introduced, using a general kernel and shell structure and its particular instances: α-clique structured graphs of connections and a hub and spoke transformation of the source graph. These structures enable the concentration and ordering of transport between vertices and reduction of the analyzed graph. To obtain the desired structures, several versions of a specialized evolutionary algorithm were developed and applied.

Keywords: *transport, connections graph, hub and spoke, kernel and shell, α-clique, genetic algorithm*

1. Introduction

A logistic system is usually represented by a graph of connections, allowing the introduction of corresponding kernel and shell structures [9, 10]. The vertices of this graph can be railway stations, bus stations or airports, etc., and edges correspond to the presence of connections between vertices. In this paper, we propose an evolution-

*Warsaw School of Information Technology, ul. Newelska 6, 01-447 Warsaw, Poland, e-mail: kulma@ibspan.waw.pl

**Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, ul. Newelska 6, Poland, e-mail addresses: Jan.Owsinski@ibspan.waw.pl, sep@ibspan.waw.pl, stanczak@ibspan.waw.pl

ary method for optimizing a logistic network by introducing a kernel and shell structure, which is a generalization of the well known hub and spoke structure, [3, 4, 8], and similar approaches, including the α-clique structure, [6, 7, 9–11].

The kernel and shell structure enables convenient concentration of flows of transport. The kernel subgraph is constituted of a set of strongly connected vertices with cheap, fast or frequent connections (depending on the transportation system modeled), while the shell vertices are less frequently connected, mostly with their kernel vertices (hubs). Instead of many bilateral connections between vertices, only local connections between kernel vertices and local connections between kernel vertices and the corresponding shell vertices are required. To simplify the problem, we deal with simple, undirected and unweighted graphs but the methods proposed can be extended and applied to weighted and directed graphs. The graph of an existing traffic system has vertices corresponding to traffic nodes and edges corresponding to traffic connections. The kernel and shell structure reduces the complexity of the resource management problem and allows more frequent connections between selected points, lower average journey times, lower costs of transport, and a lower number of vehicles required to service all connections. Thanks to this transformation, local connections are easier to synchronize and it is easier to make timetables. We use two basic approaches to transform an unstructured graph of connections into a kernel and shell structure: the α-clique method, [13], and the hub and spoke method. In the case where connections between shell nodes and their kernel node and also bilateral connections among nodes within the shell structure are required, the α-clique seems to be an appropriate solution, [12, 13]. This method is based on the notion of an α-clique, a connected subgraph with nodes less connected than a clique, but with at least an α percentage of connected nodes within an α-clique. This method, and the notion of an α-clique were developed by the authors as a method of transforming the graph of connections into a kernel and shell structure. The α-clique method can be applied in two modes:

- the number of α-cliques and possibly some kernel nodes are predetermined,
- values of α for the kernel subgraph and the shell subgraphs are imposed and the kernel nodes are chosen using an appropriate method.

The first case is probably more useful than the second, because in real-world systems candidates for kernel nodes are often known. The hub and spoke method has been developed based on work presented in [3, 4, 8], where similar structures and their applications are described. This method is useful in situations where only connections between the kernel node (the hub) and the shell nodes (spokes) are important, all the remaining transfers are realized via kernel nodes. Similarly, as in the case of the α-clique method, it is possible to define several cases of applications:

- predetermined number of and/or specified kernel nodes,
- minimum number of kernel nodes assuring the connectivity of a graph,
- indirectly determined structure of connections according to chosen parameters.

The evolutionary algorithm (EA) has been chosen as a tool for transforming such graphs, because the transformation process is a hard computational problem and there are no efficient algorithms specifically designed for solving it.

2. Graphs

The notions described below are based on [15]. 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 $v_1 \neq v_2$. A clique (a complete subgraph) $Q = (V_q, E_q)$ in the graph $G = (V, E)$ is a graph such that $V_q \subseteq V$ and $E_q \subseteq E$ and $\mathrm{Card}(V_q) = 1$ or each pair of vertices $v_1, v_2 \in V_q$ fulfills the condition $\{v_1, v_2\} \in E_q$ [2]. Any subgraph of a clique is also a clique.

An α-clique [6, 10–12] can be defined as follows: let $A = (V', E')$ be a subgraph of graph $G = (V, E)$, $V' \subseteq V$, $E' \subseteq E$, $k = \mathrm{Card}(V')$ and let k_i be the number of vertices $v_j \in V'$ such that $\{v_i, v_j\} \in E'$.

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

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

From here onwards, we will use the notion of α-clique to denote α-clique(α) for an earlier established α. A subgraph of an α-clique may not be an α-clique for the established α.

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

- kernel – a subgraph, which is constituted of a group of strongly connected vertices $K(V_k, E_k)$, depending on the needs and possibilities of the transport system or on the structure of the input graph, it can be a clique (ideally), an α-clique, or a connected subgraph;
- shell – a graph $S(V_s, E_s)$ where $V_s = V - V_k$ and $E_s = E - E_k$, depending on the requirements of the optimized transportation system, it can be an α-clique (including its kernel node) or a tree with the kernel node being the root and shell nodes being leaves.

For logistic modeling, we propose evolutionary methods (denoted EA, for evolutionary algorithms) that transform the graph of connections into an instance of the shell and kernel structure leading to hub-and-spoke or α-clique structures according to problem-specific restrictions. An α-clique structure in the graph of connections (Fig. 1b) is also an instance of the more general kernel and shell form. It consists of several peripheral (shell) α-cliques G_α with the desired values of α, connected with a central (kernel) α-clique G_c of strongly connected nodes, ideally with $\alpha \approx 1$. In the case of a sparse graph of connections, the requirement that $\alpha \approx 1$ for the kernel subgraph can be weakened to the condition that it must be a connected graph to preserve its functionality. Depending on the type of solution required, the kernel nodes or the number of kernel nodes can be imposed, or the evolutionary method can suggest the best can-

didates, taking into account the α parameters imposed on the shell α-cliques, which describe the strength of connections within the derived α-cliques. In both cases, the EA maximizes the strength of connections within the obtained α-cliques and tries to derive structures with the desired properties.

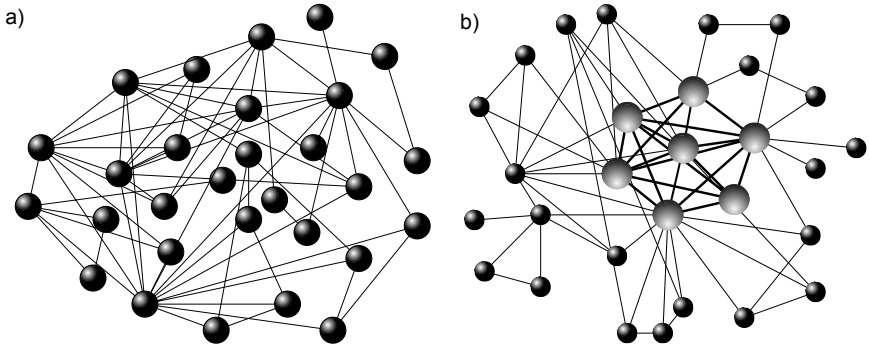

Fig. 1. A source graph (a) and the α-clique kernel and shell structure obtained (b)

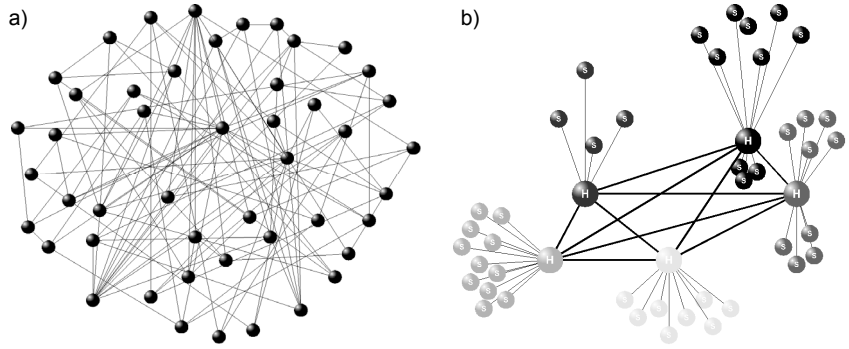

Fig. 2. A source graph (a) and the hub and spoke structure obtained (b)

A hub and spoke structure (Fig. 2b) is a graph $H_s = (G_h \cup G_s, E)$, where the subset G_h forms at least a connected graph (kernel) with the relevant subset of edges from the set E, each vertex from the subset G_s has degree 1 and is connected with exactly one vertex from the subset G_h (shell) [7, 12]. Hub and spoke is a particular case of the kernel and shell structure. This structure can be used in logistic models, where direct connections between the nodes-"spokes" attached to a 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, given by some expert, number of communication hubs with the possibility of directly determining which nodes should become hubs or selecting them by an appropriate method. The second method makes an attempt to find the minimum number of hubs which constitute at least a con-

nected subgraph with all the remaining nodes connected to their hubs. It should be noted that the number of hubs used in the first method must be at least as big as this minimum value. The third method assumes that the number of communication hubs is determined indirectly by the program parameters imposed, mainly the value of α (a hub subgraph must constitute an α-clique with the imposed value of α).

3. The evolutionary method of finding the kernel and shell structure of the graph of connections

EAs are often used to solve difficult graph problems such as graph coloring, TSP, graph partitioning, finding maximum cliques, etc., [1, 5, 14, 16], thus it seems fully justified to apply an evolutionary algorithm to the presented problem. The problem of encoding (an individual representation) depends on the desired graph structure to be obtained using the EA. In our approach, the information about any graph being processed is stored in an adjacency matrix describing all the connections between nodes of the graph, but particular problems require different methods to represent solutions. The "kernel and shell" representation of a graph of connections is a general structure, several particular instances of which are described in this article together with a specific method of encoding in each case. Generally, the encoding methods described are rather similar, but it is necessary to point out the differences. The fitness function is a modified (scaled, translated, etc.) function evaluating each member of the population (how good a solution each represents). It is calculated for computational purposes within the EA. This quality function is responsible for obtaining the appropriate structure of a graph. This function must precisely direct the EA toward the desired structure of the graph, but usually several quality functions may be used, depending on the input data, or the set of kernel nodes or shell nodes one wants to obtain. Usually, the fitness function has to contain a penalty component for potentially invalid solutions. Another important problem is to design genetic operators for the accepted data structure, taking into account the constraints imposed on solutions, so that when standard crossover and mutation operators are not appropriate, specialized operators must be defined to efficiently solve such problems. This section describes appropriate modifications of the methods of solution in detail.

3.1. The α-clique structure of kernel and shell

3.1.1. The α-clique structure with a predetermined number of kernel nodes

The data structure accepted contains a table of the predetermined number of α-cliques which constitute a shell of ordinary nodes. Each α-clique has a list of nodes attached and the element chosen to be the representative of this α-clique in the kernel

structure – a communication hub. Each node is considered only once in each solution, thus the α-cliques are disjoint structures. The kernel is an α-clique with the value of α as big as possible. Ideally, a kernel should constitute a clique, but in real cases where the connections between nodes are rather sparse, it is admissible that the kernel constitutes simply a connected graph. This condition is checked during the computations and if it is not satisfied, the penalty function significantly decreases the value assigned to the quality of such a solution:

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

where: n – imposed number of shell α-cliques in the solution evaluated, k – 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 of the ith α-clique and other nodes in this α-clique, m – number of connected subgraphs in the kernel, $\alpha_{\min}$ – minimum value of α in the shell α-cliques derived, α_K – value of α in the kernel α-clique derived.

The fitness function (1) promotes a kernel α-clique with a value of α_K as close to 1 as possible, or if the transformed graph is very sparse favors a connected subgraph, if this is possible. If not, this means that the input graph is not connected and the problem is unsolvable. The shell α-cliques should have values of α as high as possible and as many as possible of their nodes should be connected with their communication hub. The data structure described requires specialized genetic operators, which modify the population of solutions. Each operator is designed to preserve the property of being α-clique in the modified parts of solutions. If a modified solution violates the condition of being an α-clique, then the operation is canceled and no modification of the solution is performed. Using this method, it is more difficult for the evolutionary algorithm to find satisfactory solutions, due to the possible bigger problems with existence of local maxima than when using the penalty function, but it gives the certainty that the solutions computed are always feasible. The genetic operators defined are:

1. Mutation – the exchange of randomly chosen nodes in different α-cliques.
2. Movement of a randomly chosen node to a different α-clique.
3. Random selection of a kernel node from a selected α-clique – this operator is inactive when kernel nodes are explicitly assigned.
4. "Intelligent" movement – performed only if such a modification gives a better value of the fitness function.

3.1.2. The α-clique structure with a non-fixed number of kernel nodes

In the case of a non-fixed size of kernel subgraph, the number of kernel nodes is unknown and varies during the computations. Instead of fixing the number of shell

nodes, the minimum values of α for the shell α-cliques are imposed on accepted solutions. This limitation, together with the quality function to be optimized (2), indirectly determines the structure of the solutions obtained. Thus, the algorithm must find the best candidates for kernels. In this case, a somewhat different method of encoding is accepted than in the previous case. The data structure contains a list of the kernel nodes chosen (communication hubs) and lists of α-cliques, which constitute shells of ordinary nodes. As in the previous case, each node is considered only once in each solution.

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

where: n – number of α-cliques in the solution evaluated, m – number of connected subgraphs in the kernel subgraph, k_i – number of nodes in the ith α-clique, k – number of nodes in the whole graph, l_i – number of connections between the hub from the ith α-clique and other nodes in this α-clique, h_i – number of connections between hub i and other hubs.

The fitness function (2) promotes lower numbers of bigger shell subgraphs, ideally of a size almost equal to the average number of nodes in α-cliques, thus minimizing the number of α-cliques obtained, while assuring the connectivity of the kernel subgraph, maximizing the number of connections between hubs and the number of connections between each hub selected and its α-clique. As in the previously described case, each operator is designed in such a manner that the modified parts of these solutions remain α-cliques. If the modified part of a solution is not an α-clique, then the operation is canceled and no modification of the solution is performed. Additionally, each operator modifies the elements selected to be hubs for each α-clique, using a simple mutation operator. The genetic operators defined are:

1. Mutation – exchange of randomly chosen nodes in different α-clique.
2. Movement of a randomly chosen node to a different α-clique.
3. "Intelligent" movement – performed only if this modification gives a better value of the fitness function;
4. Concatenation – attempt to concatenate (mainly small) α-cliques.

3.2. The hub and spoke structure

3.2.1. The hub and spoke structure with a predetermined number of kernel nodes

A population member defines the *spokes* that do not constitute α-cliques, but are 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 real applications where hubs are defined to be the existing junction nodes, it is admissible that the subgraphs of hubs simply constitute a connected graph. The kernel nodes can be explicitly assigned or only their number may be set. An individual encoding resembles the structures used in the previously described cases, containing a table of the hubs selected with the corresponding, dynamic lists of selected spokes for each hub. For the hub and spoke structure with a predetermined number of kernel nodes, the quality function promotes solutions where a rather small subgraph of hubs is (almost) fully connected and the generated sets of spokes attached to their hubs are of medium size. The fitness function (3) promotes bigger shell subgraphs, ideally of a size almost equal to the average number of spokes, assuring the connectivity of the kernel and maximizing the number of connections among hubs:

$$\max Q = \frac{1}{n}\sum_{i=1}^{n}\left(k_i - \left| \frac{k-n}{n} - k_i \right| + \frac{h_i}{n} \right) \tag{3}$$

where: n – predetermined number of hubs in the solution evaluated, m – number of connected subgraphs in the kernel subgraph, k_i – number of spokes attached to the ith hub, k – number of nodes in the whole graph, h_i – number of connections between hub i and other hubs.

This problem can be solved using similar operators to the ones used for the α-clique methods, but different conditions are checked before they are performed. When one node (spoke) is to be moved to another hub shell, first it must be checked that it is connected to this new hub. If not, the operation is canceled with similar consequences to those previously described for the α-clique method. In this case, the set of genetic operators contains:

1. Mutation – 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 hubs of randomly selected spokes – this operator is inactive when kernel nodes are explicitly assigned.

A problem arises when the predetermined number of kernel nodes is lower than the minimal value that assures that all the shell nodes will be attached to their kernel hubs. This problem can be solved in two ways. The former enables the final result to contain unattached shell nodes. The other increases the number of kernel nodes to obtain a connected graph. These methods are realized using modified forms of the quality function (3) with a penalty for unattached shell nodes or additional kernel nodes.

3.2.2. The hub and spoke structure with a non-fixed number of kernel nodes

In this case, the number of kernel nodes (hubs) is unknown in advance and varies during the computations. The algorithm must find the set of kernel candidates optimiz-

ing the quality function (4). The data structure contains a dynamic table of the kernel nodes chosen and dynamic lists of spokes, which constitute shells of ordinary nodes. As in the previously considered cases, each node is considered only once in each solution.

$$\max Q = \frac{1}{nm}\sum_{i=1}^{n}\left(k_i - \left|\frac{k-n}{n} - k_i\right| + \frac{h_i}{n}\right) \tag{4}$$

where: n – number of hubs in the solution evaluated, m – number of connected subgraphs in the kernel subgraph, k_i – number of nodes (spokes) attached to the ith hub, k – number of nodes in the whole graph, h_i – number of connections between hub i and other hubs.

The fitness function (4) promotes bigger shell subgraphs, ideally of a size almost equal to the average number of spokes, assuring connectivity of the kernel subgraph and maximizing the number of connections among hubs. The set of genetic operators in this case contains:

1. Mutation – 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" versions, the "intelligent" version performs changes in an individual only if the new set of spokes is better connected with their kernels than before this operation).
3. Exchange of a randomly selected hub for a randomly selected spoke (as in the previous case – random and the "intelligent" versions).
4. Concatenation – attempt to concatenate two sets of spokes (so as to minimize the number of kernel nodes).

3.2.3. 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 an indirectly imposed number of kernel nodes but this problem is computationally more difficult to solve. The method of encoding is identical to the case of a non-fixed size of kernel subgraph but the fitness function to be optimized is different:

$$\min Q = nm \tag{5}$$

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

The fitness function (5) promotes small sets of connected kernel nodes with all the spokes attached to their hubs. The set of genetic operators in this case contains:

1. Mutation – 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" versions, the "intelligent" version performs changes in an individual only if the new set of spokes is better connected with their kernels than before this operation).

3. Exchange of a randomly selected hub for a randomly selected spoke (as in the previous case, random and "intelligent" versions).

4. Concatenation – attempt to concatenate two sets of spokes (tries to minimize the number of kernel nodes).

4. Results of computer simulations

To test these methods, we used examples from BHOSLIB (Benchmarks with Hidden Optimum Solutions for Graph Problems):

http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm.

The problems selected involve a graph with 450 vertices and 83 198 edges with the maximum clique size equal to 30 (frb30-15-clq.tar.gz) and a graph with 4000 vertices and 7 425 226 edges (frb100-40.clq.gz). The sizes of these problems are relatively big, but their complexity is similar to problems encountered during the planning of real connections.

4.1. Results obtained for the α-clique method

4.1.1. The problem with a non-fixed number of kernel nodes

The first step of obtaining a kernel and shell structure from the input graph using the α-clique method is to find a value of α that gives the desired results. Thus, it is necessary to decide which solution is the closest to our requirements. Using various values of α, we obtained several different solutions. It is difficult to foresee a priori, taking into account only the value of α, which value would be the best, but after computations it is quite easy to choose the solution with the most acceptable parameters. The most important factor influencing this decision is the number of kernel nodes obtained. For a problem similar to that of connecting large European cities (frb30-15-1), the best solutions are probably the ones for $\alpha = 0.8$ with 10 kernel nodes. Of course, we can also try to find different numbers of kernel nodes by adopting values of α between the tested ones. The process of genetic computations lasts about 5–10 minutes, depending on the operating system and machine used, together with the value of α. Thus it is possible to compute solutions using several values of α, before accepting one. The results presented provide a distinctly simplified structure of connections. For example, in the case of $\alpha = 0.80$, we obtained an output structure with about 1018 connec-

tions in place of the input structure with 83 198 connections. This lower number of connections implies faster and more frequent connections with lower costs for carriers and their clients. The results obtained for larger graphs have similar properties, but the computations last longer, especially for the frb100-40 problem. The computational complexity depends mainly on the number of edges, the number of vertices is less significant. It should be noted that for all the graphs considered, the kernel constitutes at least a connected subgraph with strongly connected α-cliques attached to shell subgraphs.

Table 1. Comparison of the results obtained with various values of α

α	Subject	Min	Max	Median	No. of kernel nodes	α_K	No. of connections
Name of problem: frb30-15-1		Vertices: 450			Edges: 83 198		
0.80	cardinality of shell	40	50	45	10	1.00	9067
	degree of kernel nodes in shell subgraphs	39	48	43			
	degree of kernel nodes in kernel subgraph	10	10	10			
0.90	cardinality of shell	22	23	22	20	0.95	5225
	degree of kernel nodes in shell subgraphs	21	23	22			
	degree of kernel nodes in kernel subgraph	19	20	20			
1.00	cardinality of shell	14	16	15	30	0.93	4025
	degree of kernel nodes in shell subgraphs	14	16	15			
	degree of kernel nodes in kernel subgraph	28	30	29			
Name of problem: frb100-40		Vertices: 4000			Edges: 7 425 226		
0.80	cardinality of shell	4000	4000	4000	1	1.00	7 429 226
	degree of kernel nodes in shell subgraphs	3727	3727	3727			
	degree of kernel nodes in kernel subgraph	1	1	1			
0.90	cardinality of shell	76	420	187	20	0.75	462 250
	degree of kernel nodes in shell subgraphs	74	407	173			
	degree of kernel nodes in kernel subgraph	16	20	18			
1.00	cardinality of shell	44	49	48	82	0.96	102 869
	degree of kernel nodes in shell subgraphs	44	49	48			
	degree of kernel nodes in kernel subgraph	79	82	81			

4.1.2. The problem with a fixed number of kernel nodes

In this approach, the results depend on the number of shell structures. The results obtained show that conversion of a graph of connections into a kernel and shell structure can be obtained using a different approach. As in the previous case, the computations last significantly longer for the graph with 7 425 226 edges but the kernel subgraphs are successfully connected. All the constraints imposed on the obtained kernel and shell structures can be fulfilled for a range of numbers of kernel nodes,

beyond these values it would be necessary to modify the graph of connections to obtain feasible solutions.

Table 2. Comparison of the results obtained for various numbers of kernel nodes

Desired number of kernel nodes	Subject	Min	Max	Median	$\alpha_{\min}$	α_K	No. of connections
Name of problem: frb30-15-1		Vertices: 450		Edges: 83 198			
5	cardinality of shell	53	140	92	0.81	1.00	20 035
	degree of kernel nodes in shell subgraphs	45	117	79			
	degree of kernel nodes in kernel subgraph	5	5	5			
10	cardinality of shell	28	78	34	0.82	1.00	11 140
	degree of kernel nodes in shell subgraphs	26	67	32			
	degree of kernel nodes in kernel subgraph	10	10	10			
50	cardinality of shell	1	21	7	1.00	0.80	5111
	degree of kernel nodes in shell subgraphs	1	21	7			
	degree of kernel nodes in kernel subgraph	40	49	42			
100	cardinality of shell	1	20	1	1.00	0.79	7881
	degree of kernel nodes in shell subgraphs	1	20	1			
	degree of kernel nodes in kernel subgraph	79	93	84			
Name of problem: frb100-40		Vertices 4000		Edges 7 425 226			
5	cardinality of shell	230	1871	537	0.90	1.00	2 319 961
	degree of kernel nodes in shell subgraphs	211	1727	502			
	degree of kernel nodes in kernel subgraph	5	5	5			
10	cardinality of shell	46	1706	126	0.91	1.00	1 917 039
	degree of kernel nodes in shell subgraphs	42	1558	120			
	degree of kernel nodes in kernel subgraph	10	10	10			
50	cardinality of shell	44	538	59	0.93	0.94	322 273
	degree of kernel nodes in shell subgraphs	41	503	56			
	degree of kernel nodes in kernel subgraph	47	50	48			
100	cardinality of shell	1	71	49	1.00	0.89	120 893
	degree of kernel nodes in shell subgraphs	1	98	49			
	degree of kernel nodes in kernel subgraph	89	98	93			

4.2. Results obtained for the hub and spoke method

4.2.1. The problem of finding the minimum number of kernel nodes

It should be noted that this version of the algorithm limits the possibilities of transforming the graph too strongly. In this case, only one solution is generated and the sets of spokes may be too big to be useful. But this result helps us to find the lower limit on the possible and practically useful numbers of hubs.

Table 3. The minimum numbers of kernel nodes

No. of kernel nodes obtained	Subject	Min	Max	Median	α_K	No. of connections
Name of problem: frb30-15-1		Vertices: 450			Edges: 83 198	
2	cardinality of shell	104	344	104	1.00	899
	degree of kernel nodes in shell subgraphs	104	344	104		
	degree of kernel nodes in kernel subgraph	2	2	2		
Name of problem: frb100-40		Vertices: 4000			Edges: 7 425 226	
2	cardinality of shell	435	3563	435	1.00	7999
	degree of kernel nodes in shell subgraphs	435	3563	435		
	degree of kernel nodes in kernel subgraph	2	2	2		

4.2.2. The problem with a fixed number of kernel nodes

This method generates a kernel and shell structure with the given number of hubs. Due to the structure of the input graph, some hub and spoke structures with a predefined number of kernel nodes may not be achieved without leaving any shell nodes unattached. In the results presented here, we considered the results obtained in the previous section and requested kernel sizes bigger than the minimal values obtained.

Table 4. Results obtained for various desired numbers of hubs

The desired number of kernel nodes	Subject	Min	Max	Median	α_K	No. of connections
Name of problem: frb30-15-1		Vertices: 450			Edges: 8319	
5	cardinality of shell	89	89	89	1.00	901
	degree of kernel nodes in shell subgraphs	89	89	89		
	degree of kernel nodes in kernel subgraph	5	5	5		
10	cardinality of shell	44	44	44	1.00	926
	degree of kernel nodes in shell subgraphs	44	44	44		
	degree of kernel nodes in kernel subgraph	10	10	10		
50	cardinality of shell	8	8	8	0.92	1975
	degree of kernel nodes in shell subgraphs	8	8	8		
	degree of kernel nodes in kernel subgraph	44	50	46		
100	cardinality of shell	3	4	3	0.89	5232
	degree of kernel nodes in shell subgraphs	3	4	3		
	degree of kernel nodes in kernel subgraph	89	96	92		
Name of problem: frb100-40		Vertices: 4000			Edges: 7 425 226	
5	cardinality of shell	799	799	799	1.00	8001
	degree of kernel nodes in shell subgraphs	799	799	799		
	degree of kernel nodes in kernel subgraph	5	5	5		

The desired number of kernel nodes	Subject	Min	Max	Median	α_K	No. of connections
10	cardinality of shell	399	399	399	0.90	8024
	degree of kernel nodes in shell subgraphs	399	399	399		
	degree of kernel nodes in kernel subgraph	9	10	10		
50	cardinality of shell	79	79	79	1.00	9126
	degree of kernel nodes in shell subgraphs	79	79	79		
	degree of kernel nodes in kernel subgraph	50	50	50		
100	cardinality of shell	39	39	39	0.99	12 717
	degree of kernel nodes in shell subgraphs	39	39	39		
	degree of kernel nodes in kernel subgraph	99	100	99		

The results from Table 4 show that the method based on a given number of hubs is more flexible, because it is possible to obtain the desired structure of the transformed graph, while the previous method gives only one solution for each case. As it can be seen, for larger numbers of hubs, the subgraph becomes not fully connected, but the numbers of connections between hubs are very high (α is close to 1). However, it is possible to obtain worse results for sparse graphs or for a larger number of hubs. It is thus necessary to assess at least the connectivity of the kernel subgraph.

4.2.3. The problem with a non-fixed number of kernel nodes

The results obtained in this case are similar to those obtained in the case of minimum kernel size (see Section 4.2.1, Table 3), although this is not a general rule. We conducted simulations for different sets of data and the results obtained were, on occasion, different.

5. Conclusions

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

References

[1] CHEN Z.-Q., WANG R.-L., OKAZAKI K., *An Efficient Genetic Algorithm Based Approach for the Minimum Graph Bisection Problem*, IJCSNS International Journal of Computer Science and Network Security, 2008, 8 (6), 118–124.

[2] CORMEN T.H., LEISERSON CH.E., RIVEST R.L., STEIN C., *Introduction to Algorithms*, 3rd Ed., MIT, Cambridge, Mass., USA, 2009.

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

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

[5] MARCHIORI E., *A Simple Heuristic Based Genetic Algorithm for the Maximum Clique Problem*, Proceedings of the 1998 ACM Symposium on Applied Computing, 1998, ACM, 366–373.

[6] MAŻBIC-KULMA B., POTRZEBOWSKI H., STAŃCZAK J., SĘP K., *Evolutionary approach to solve hub-and-spoke problem using α-cliques*, [in:] *Evolutionary Computation and Global Optimization*, Prace Naukowe PW, Elektronika, Warsaw 2008, 165, 121–130.

[7] MAŻBIC-KULMA B., POTRZEBOWSKI H., STAŃCZAK J., SĘP K., *Evolutionary approach to find kernel and shell structure of a connection graph*, TLM AGH, Cracow 2009, 37–50.

[8] MIN H., GOU Z., *The location of hub-seaports in the global supply chain network using a cooperative competition strategy*, Integrated Supply Management, 2004, 1 (1), 51–63.

[9] POTRZEBOWSKI H., STAŃCZAK J., SĘP K., *Evolutionary algorithm to find graph covering subsets using α*-cliques, [in:] *Evolutionary Computation and Global Optimization*, J. Arabas (Ed.), Prace Naukowe PW, Konferencje, Warsaw 2006, 351–358.

[10] POTRZEBOWSKI H., STAŃCZAK J., SĘP K., *Heurystyczne i ewolucyjne metody znajdowania pokrycia grafu, korzystające z pojęcia alfa-kliki i innych ograniczeń. Badania operacyjne i systemowe 2006. Metody i techniki*, Akademicka Oficyna Wydawnicza EXIT, Warsaw 2006.

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

[12] POTRZEBOWSKI H., STAŃCZAK J., SĘP K., *Evolutionary approach to solve hub-and-spoke problem using α-cliques*, *Evolutionary Computation and Global Optimization*, Prace Naukowe PW, Warsaw 2008, 165, 121–130.

[13] STAŃCZAK J., POTRZEBOWSKI H., SĘP K., *Evolutionary approach to obtain graph covering by densely connected subgraphs*, Control and Cybernetics, 2011, 41 (3), 80–107.

[14] TALBI E.-G., BESSIERE P., *A parallel genetic algorithm for the graph partitioning problem*, Proceedings of the 5th International Conference on Supercomputing, ACM, New York 1991, 312–320.

[15] WILSON R.J., *Introduction to Graph Theory*, Longman, Harlow 1996.

[16] YU T.L., GOLDBERG D.E., YASSINE A., YASSINE C.A., *Genetic algorithm design inspired by organizational theory*, Genetic and Evolutionary Computation Conference, Chicago, Illinois, USA, Springer-Verlag, Lecture Notes in Computer Science, Heidelberg 2003.