DOI: 10.15439/2015F218
ACSIS, Vol. 5

Graph based approach to the minimum hub problem in transportation network

J. Owsiński, J. Stańczak, A. Barski, K. Sęp
Systems Research Institute,
Polish Academy of Sciences
ul. Newelska 6,
01-447 Warszawa, Poland
Email: {owsinski, stanczak, Aleksy.Barski, sep} @ibspan.waw.pl

P. Sapiecha
The Faculty of Electronics
and Information Technology,
Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: {sapiecha@tele.pw.edu.pl}

Abstract*—*In this paper we consider a hub location problem in a real multimodal public transportation network. This problem is also known as the park-and-ride problem. Hubs stations are special facilities that serve as switches in such a network. In practice the set of hubs has a strategic importance, because all of the traffic that passes through the network can be controlled by these elements. From the theoretical point of view, the minimal hub problem is NP hard. Two different approaches to this problem are presented. The first group of methods bases on the greedy algorithms. In the second group the evolutionary strategy is used. The computational results for these algorithms proved a significant efficiency, what can be clearly expressed in terms of an input data reduction and also in quality measure values for the obtained solutions of this problem.

I. Introduction

LET US consider a real public transportation network. This network can be described as a graph [3], [8], [9]. In this graph, each tram/bus stop corresponds to one vertex and any two vertices are adjacent iff they belong to at least one common public transport line. Hence, the set of vertices along a route forms a connected subgraph of the entire graph. Each vertex in this graph is characterized/labeled by a set of numbers of tram/bus lines passing through it.

In such a graph, a hub set is a subset of vertices, such that any two vertices are connected by a path whose vertices lie in it. Let us define the minimum hub set problem as the problem of finding the hub set of minimal cardinality for a given graph. This collection has a strategic importance for the transportation system, because all the traffic that passes through the network can be controlled by these vertices. Communication hubs are excellent candidates for park-and-ride (P&R, sometimes P+R) points ([4]) with good connections to the center and other parts of the city and this idea is a basis of presented work. This problem is well known and is commonly used in industry, in particular in such areas as transport, telecommunication, and distributed computing [1], [2], [11], [13], [16].

Let us observe that we can try to find the minimum hub set for a given network in two different ways. This problem would be directly reduced to the task of designation the minimum dominating set [6], [14], [15]. Alternatively, we have a possibility of creating a hypergraph, such that its set of vertices is the same as in the original network, and for each set of vertices, which correspond to stops lying along a tram/bus route, we form the hyperedge. In this case, we reduce the considered problem to the search for the minimal transversal in a constructed hypergraph.

II. Basic mathematical definitions

This section provides some basic notation. A **graph** is a representation of a set of objects, where some pairs of objects are connected by links. The interconnected objects are represented by mathematical abstractions called vertices, and the links that connect some pairs of vertices are called edges. More formally, a **graph** is an ordered pair G=(V, E) comprising a set V of **vertices** or **nodes** together with a set E of **edges**, which are 2-element subsets of V (E is a subset of VxV). An **undirected** graph is the one in which edges have no orientation. The edge (a, b) is then identical to the edge (b, a).

Let $N(v)=\{u \in V: (v, u) \in E\}$ be an open **neighborhood** for a given vertex v.

A **dominating set** for a graph G is a subset D of V such that every vertex not in D is adjacent to at least one member of D. This problem is strongly related to a problem well known in computational geometry, the **art gallery problem.** The **domination number** $\gamma(G)$ is the number of vertices in a smallest dominating set for G. The k-**dominating set problem** concerns testing whether $\gamma(G) = k$ for a given graph G and natural number k; it is a classical NP-complete decision problem in computational complexity theory (Garey & Johnson 1979) [3], [8], [9].

Theorem (Ore): If G=(V, E) is a graph without isolated vertices, then the complement of a minimal dominating set of G is also a dominating set of G. This implies that every such graph has two disjoint dominating sets and hence, $\gamma(G) \leq ½ \,\mathrm{Card}(V)$ [3], [8], [9].

Theorem (Arnautov 1974, Payan 1975, Alon 1990) : If G=(V, E) is a graph with minimum degree $d > 1$, then $\gamma(G) \leq [(1 + \ln(d+1))/(d + 1)]\, \mathrm{Card}(V)$ [3], [8], [9].

An **independent set** is a set of vertices in a graph, such that no two of them are adjacent. The size of such a set is called the **independence number** of G, and is denoted α(G). The problem of finding the set such that α(G) = k is called the k-**independent set problem** and is an NP-complete decision problem. The dominating sets are closely related to independent sets (e.g. the 8-**Queens Problem** Puzzle). Namely, an independent set is also a dominating set if and only if it is a maximal independent set, so any maximal independent set in a graph is necessarily also a minimal dominating set.

A **hypergraph** is a generalization of a graph in which an edge can connect any number of vertices. Formally, a hypergraph H is a pair H=(X,F), where X is a set of elements called **vertices**, and F is a set of non-empty subsets of X called **hyperedges**. Let F be a subset of P(X)\{Ø}, where P(X) is the power set of X and F(x) = {f ∈F: x∈f } for x ∈ X. A hypergraph is also called a **set system** or a family of sets drawn from the universal set X. The **rank** of hypergraph H is the size of the largest hyperedge in H. A **set covering** of a hypergraph H=(X, F) is a subfamily C of F, such that the union of hyperedges from C equals the universe of vertices. A **transversal** (or **hitting set**) of a hypergraph H=(X, F) is a subset T of X that has a nonempty intersection with every edge. The notions of hitting set and set covering are equivalent. The **decision versions** of **hitting set** and **set covering** problems are NP-complete.

For any given graph G=(V, E) with V={1, 2,..., n}, construct a hypergraph H=(X, F) as follows: the universe X is V, and the family of hyperedges F is {F1, F2, ..., Fn} such that Fv consists of the vertex v and all vertices adjacent to v in G. Hence, if D is a dominating set for G, then S={Sv: v∈D} is a feasible solution of the set cover problem, with Card(C)=Card(D). Conversely, if S={Sv: v∈D} is a feasible solution of the set cover problem, then D is a dominating set for G, with Card(D)=Card(C).

The **greedy algorithm** for set covering chooses the sets according to one rule: at each stage, choose the hyperedge that contains the largest number of uncovered elements. This algorithm actually achieves an **approximation ratio** Card(C)/Card(Opt) (Opt – is an optimal set covering) of **h(rank)**, where h(n) is the n-th harmonic number. This value is approximately given by: ***O*((*1*+ *log*(*Card*(*V*)))**. We can construct dual algorithm for hitting set problem for which performance ratio is: ***O*((*1*+ *log*(*Card*(*F*)))**.

Algorithm 0 – Greedy set covering method

```
input:  a hypergraph H=(X, F);
output:  a set covering C;
U := X; C := Ø;
while( C ≠ X )do{
    select S from F such that maximizes Card(S ∩ U);
    U := U \ S;
    C := C ∪ {S};
}
return: C;
```

It is interesting that there exists a pair of **polynomial-time reduction** between the **minimum dominating set problem** and the **minimum set covering problem** (Kannan 1992 pp.108–109). These reductions show that an efficient algorithm for the minimum dominating set problem would provide an efficient algorithm for the set cover problem and vice versa. According to the above presented facts, the greedy algorithm provides a factor 1+ log(Card(V)) approximation of a minimum dominating set.

Algorithm 1 – Greedy hitting set method

```
input:  a hypergraph H=(X, F);
output:  a hitting set – transversal T;
T := Ø; E := F;
while( E ≠ Ø )do{
    select x from X such that maximizes Card(F(x) ∩ E);
    T := T ∪ {x} ;
    E := E \ F(x);
}
    return: T;
```

Consider another approach to the hitting set problem, I.e. **algorithm 2** (**MSBT**) [12]. It seeks the vertices with the lowest degree and removes them from the set of vertices. If (without a removed vertex) the vertex set is not a transversal, then this vertex should be added to the transversal under construction, and the edges incident with it are eliminated from the hypergraph – they are deemed to be covered.

Yet another method (**algorithm RSBT**) [12] seeks a maximum transversal in the sense of cardinality. Contrary to the MSBT algorithm, the RSBT removes vertices with the biggest degree as long as it can. All the rest is the same as in the MSBT algorithm.

Algorithm 2 - MSBT

```
input:  a hypergraph H=(X, F);
output:  a hitting set – transversal T;
T := Ø V := X; Q := X; E := F;
while( V ≠ Ø & E ≠ Ø) do
{   x := vertices with the lowest degree; V := V \ {k};
    if( V is not transversal of hypergraph (Q, E) )
    then { T := T ∪ {x}; E := E \ F(x); V := V \ { v □ V: F(v) = Ø};}
      else
        for( each edge covers by exact one vertex v )do
          {T := T ∪ {v}; E := E \ F(v) ; V := V \ {v};}
    Q:=V;
}
```

The algorithms MSBT and RSBT are the algorithms designed by authors as a complement of the Algorithm 1.

Let us consider a **reduction** from the dominating set problem to the set covering problem. For any given graph G=(V, E) with V={1, 2,..., n}, construct a hypergraph H=(X, F) as follows: the universe X is V, and the family of hyperedges F is {F1, F2, ..., Fn} such that Fv consists of the vertex v and all vertices adjacent to v in G. Hence, if D is a dominating set for G, then S={Sv: v∈D} is a feasible solution of the set cover problem, with Card(C)=Card(D). Conversely, if S={Sv: v∈D} is a feasible solution of the set cover problem, then D is a dominating set for G, with Card(D)=Card(C).

According to the above presented facts, the **greedy algorithm** provides a factor 1+ log(Card(V)) approximation of a **minimum dominating set.** Additionally, Raz and Safra (1997) show that **no** algorithm can achieve an approximation factor better than clog((Card(V)) for some c > 0 unless P = NP. Fomin, Gradoni and Kratsch (2009) show an **exact** algorithm which can be used to find a minimum dominating set in time $O((1.5264)^n)$ and polynomial space. In parallel, a faster algorithm, using $O((1.5048)^n)$ time was found by van Rooij, Nederlof and van Dijk (2009).

III. Evolutionary algorithm

Graph problems such as graph coloring, TSP, graph partitioning, maximum clique search, etc. (**NP-hard optimization problems**) are often solved using computational intelligence methods due to the lack of efficient polynomial algorithms. Among them, the **evolutionary algorithms** (EAs) are often applied; thus, it seems fully justified to use the EA in the present graph transformation problem. The EA approach is quite different than the described earlier hypergraph method. The EA method tries to find hubs, which are strongly connected (in the sense of high capacity of connections – see formula (4)) among them and allow for fast mass travel to the center of the city and other hubs. It is very likely that such hubs are good candidates for the park-and-ride locations.

In our approach, information about the transformed graph is stored in an array of data describing all connections among graph nodes – public communication stops. This array is an adjacency matrix of undirected graph with stored values – weights, denoting the capacities of connections.

The idea of our approach is based on the hub and spoke paradigm. The spokes, which in this case represent public transport stops of minor importance, constitute groups of nodes connected with their hubs. The size of the subgraph of hubs is known in advance. Hubs (should) represent important public transport stops with fast and frequent connections with other important stops and the center of the city.

Algorithm 4. shows how the typical EA work, but this general framework requires several improvements to work efficiently and thus a specialized evolutionary method, developed by authors is used to solve the problem.

To adjust the EA to the solved problem it is necessary to apply proper, problem-specific **encoding** of solutions (sometimes called also population members, individuals or even agents), to develop specialized **genetic operators** tailored for the analysed data structure and the solved problem and, finally, to formulate the problem-dependent **fitness function** to be optimized by the algorithm.

Algorithm 4 - **The standard evolutionary algorithm**

```
1. input:   a given problem;
2. output:  solution of problem;
3. {    while( not stop condition )do
4.          { reproduction and modification
              of solutions using genetic operators;
5.            valuation of obtained solutions;
6.            selection of individuals for the next generation;
7.          }
8.  }
```

The first step to obtain efficient evolutionary method is to choose an appropriate encoding method. It is obvious that there are plenty of possible solutions with different advantages and drawbacks. In presented approach the solution is stored as a vector of selected hubs with lists (or variable length vectors) of attached to them spokes.

An important problem in developing an efficient EA is the design of **genetic operators** for the adopted data structure, taking into account the constraints imposed on solutions. In the case here considered, the standard crossover and mutation operators are not proper, so the problem-specific, specialized operators must be prepared to efficiently solve the problems considered. When one element of a modified solution (for instance one node) is to be moved to another place (e.g. to the cluster of another hub), it must first be checked if it has a connection with this new hub. If not, the operation is canceled to avoid creation of infeasible solutions. Altogether, here, the set of genetic operators is:

- mutation – exchange of randomly chosen nodes in different sets of spokes,
- relocation of a randomly chosen node to a different set of spokes,
- exchange of a randomly selected hub for randomly selected spoke.

Application of several specialized genetic operators requires a method of selecting and executing them in all iterations of the algorithm. In the approach used in [17] it is assumed that an operator that generated good results for some population member should have bigger probability of execution for its possible offspring and more frequently affect them than the remaining operators. But it is very likely that the operator that improves one individual and its descendants may give worse effects for other individuals, because of its location in the domain of possible solutions. Thus, every individual should have its own preferences and could be treated as an agent, whose role is to select and call one of the evolutionary operators (or to perform an action) to obtain improvement of solution (or its income) as high as possible. In the EA here used, every individual has an additional vector of floating point numbers, besides the encoded solution. Each number corresponds to one genetic operator and is a measure of its quality (a quality factor). The higher the factor, the higher the probability of calling and executing the operator. Simple normalization of the vector of quality coefficients turns it into a vector of operator execution probabilities. This set of probabilities is also an expression of experience of every individual and according to the experience gathered one can maximize the chances of its offspring to survive.

The method of computing quality factors is based on **reinforcement learning**. When the selected by the individual-agent i^{th} operator is applied, this can be regarded as an agent performing action a_i leading to a new state s_i, which, in this case, is a new solution. The **agent** (or individual, solution or population member) 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^*, maximizing its chances to have offspring in next generations:

$$V^{\Pi}=E_{\Pi}\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\right) \quad , (1)$$

$$V^*=\max_{\Pi}\left(V^{\Pi}\right) \quad . (2)$$

The following formula, derived from (1) and (2), is used for evaluation purposes:

$$V(s_{t+1})=V(s_t)+\alpha(r_{t+1}+\gamma V^*(s_{t+1})-V(s_t)) \quad , (3)$$

where:

Π	- the set of possible strategies of the agent,
V^{Π}	- the discounted cumulative reward obtained using strategy Π,
E	- expected value,
k	- consecutive time steps,
t	- current time,
$V(s_t)$	- 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	- the learning factor,
γ	- the discount factor,
r_{t+1}	- reward for the best action, equal to the improvement of the quality of solution after execution of the evolutionary operator.

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

The **fitness function** in the EA is closely connected with problem specific quality function. The fitness function evaluates the members of the population. It is a modified (scaled, translated, etc.) problem quality function, prepared for computational purposes in the EA. The quality function directs evolutionary computations to obtaining the proper graph structure. In the considered problem the quality function is formulated to direct the EA towards the desired structure of potential P&R, taking into account weights among the graph vertices. Several quality functions can be used, depending on input data (binary, integer or real) or what kind of set of P&R nodes one wants to obtain. The formula presented in this paper was obtained on the basis of experiments. Usually, such formulas contain a penalty part for the potential invalid or improper structure of the obtained solutions.

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

$$\max Q=\sum_{i=1}^{n}\left(a*\sum_{j=1}^{k_i}w_{ij}+b*w_{iC}+c*\sum_{m=1}^{k-k_i}w_{im}\right) \quad , \qquad (4)$$

where:

w_{ij}	– weight between candidate for P&R (hub) and its subgraph of communication stops (spokes),
w_{iC}	– weight between candidate for P&R (hub) and the center of the city[1],
w_{im}	– weight between candidate for P&R (hub) and remaining communication stops,
n	– predetermined (constant) number of hubs (candidates for P&R) in the solution,
k_i	– number of nodes attached to i^{th} hub,
k	– number of nodes in the whole graph,
a, b, c	– non-negative constants values that emphasize the influence of corresponding factor.

IV. Data base & computational environment

For the computational experiments a set of data files was obtained from the **www page**: http://www.ztm.waw.pl/. The description of the structure of public transport in Warsaw and full list of bus, subway and tramway stops and all lines of these means of transport are collected in the data files. The data include more than 5600 stops, but the majority of them are stops located not far from each other, with the same names but different minor numbers (like Centrum 01, Centrum 02,...), which indicate the opposite direction or different transport mean or extensions for bigger numbers of vehicles boarding passengers simultaneously. Thus, first the stops were aggregated: all of them with the same names and different minor numbers were treated as one. This allowed to reduce the size of the communication network to 1883 stops.

We prepared computation experiments for this article based on typical **personal computer** (with an Intel i5 4 x3.2 GHz microprocessor) running Linux OS. All our programs were written in the C language and compiled with the g++ compiler application.

V. Data reduction

At the second step, some preliminary computations for the considered data allowed to reduce the size of the problem even more. This **preprocessing** was oriented to **reduce** input data (filtering method). We can treat the graph G as a representation of Warsaw **transportation network** being an input data for our programs.

Looking at this graph one can observe that it is possible to remove some of the redundant edges. It is clear that if two vertices, a and b, are **adjacent** in G and the sets of bus/subway/tram lines passing through them are the same, then we can apply the edge (a, b) contraction operation. An **edge contraction** is an operation which **removes** an edge from a graph while simultaneously **merging** the two vertices that it previously joined. More precisely, the edge (a, b) is removed and its two incident vertices, a and b, are merged into a new vertex c, where the edges incident to c correspond to edges incident to either a or b. Obviously, it is possible to apply these contraction operations several times, in polynomial time. After executing this procedure an obtained graph consists of 1003 vertices (reduction of 880 vertices).

VI. The results of presented approaches

Given the above, we can try to find the **minimum hub set** for the given network by applying the presented algorithms.

[1]We decided to emphasize a central communication stop in the city as a virtual aim of the majority of commuters.

At the beginning we construct a hypergraph, such that its set of vertices is the same as it is in the input network. Let for each set of vertices, which correspond to stops lying along a tram/bus route, form a hyperedge. In this case, we **reduce** the problem considered to the **minimal transversal** construction for this hypergraph. Therefore, two different approaches are possible - deterministic and random:

(**a**) application of **greedy methods** (in different versions);

(**b**) use of **evolutionary algorithms.**

In order to compare these methods we also made computations with and without the preliminary reduction. In the first approach **(greedy method),** algorithm 2 found 35 vertices. The **RSBT** found smaller transversal – 140 vertices instead of 152 vertices found without the reduction. Considering the obtained results it looks like the reduction could be used for the EA approach. There was no change as to the result obtained in the evolutionary algorithm, but there was a significant reduction of time needed for computation.

Due to a smaller graph of public communication stops considered each iteration of EA last significantly shorter, while the number of them remained unchanged. For the greedy method the time of computation is very short (less than 1 s.) with and without the reduction, thus it is no use to compare them with EA and with or without the reduction.

TABLE 1.
COMPUTATIONAL RESULTS

Number of hubs	Time of 10 000 EA iterations (average of 10 trials)	
	Before reduction	After reduction
50	5 854 s	2 478 s
100	13 433 s	5 895 s
152	27 231 s	14 256 s

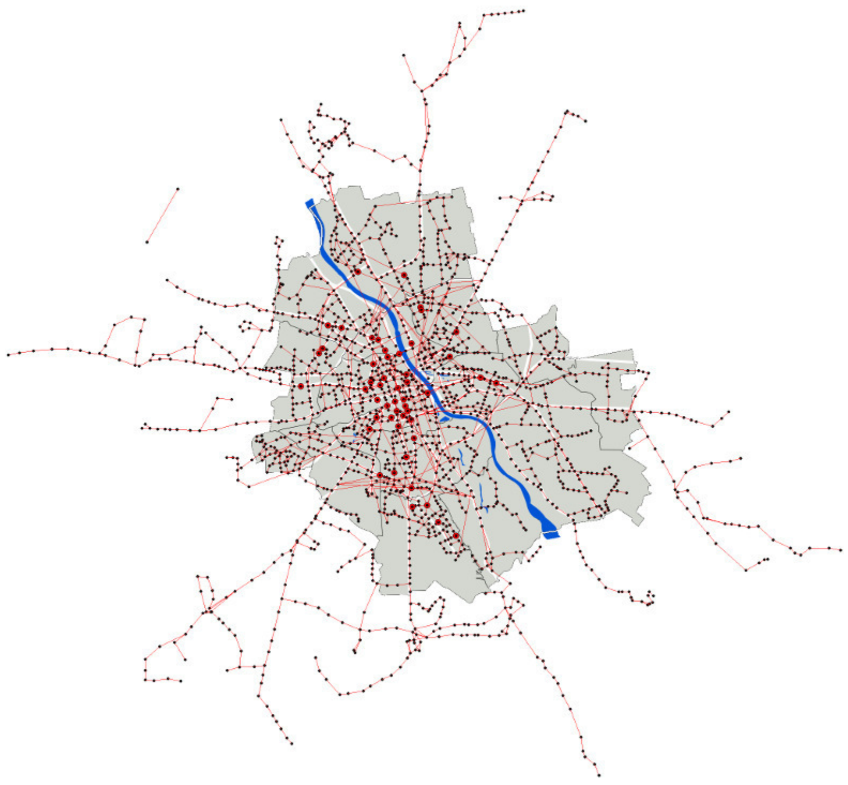

Figure 1: Result obtained by the EA with the imposed number of 50 hubs

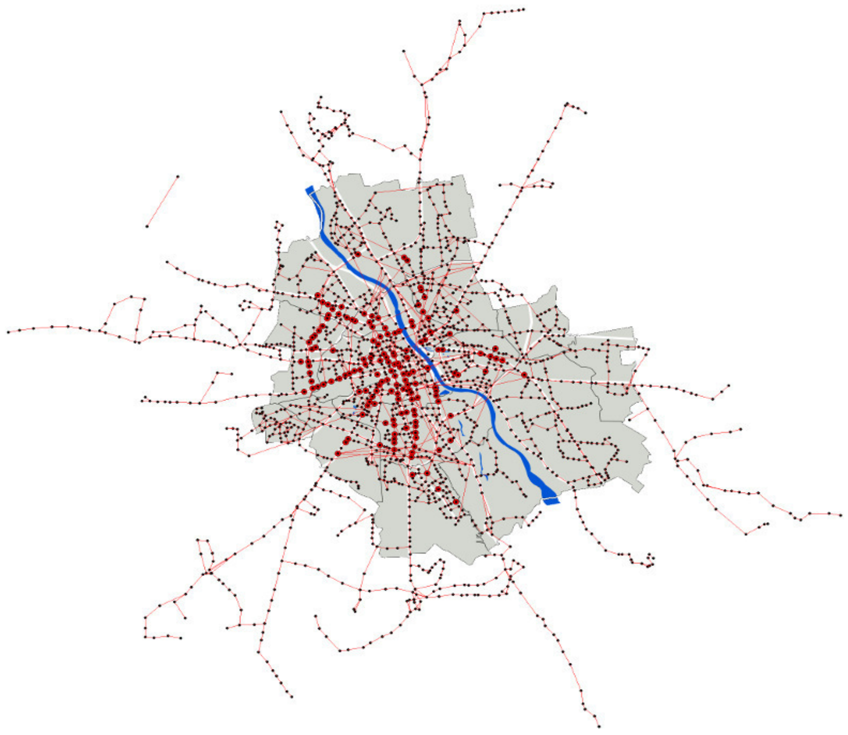

Figure 2: Result obtained by the EA with the imposed number of 152 hubs

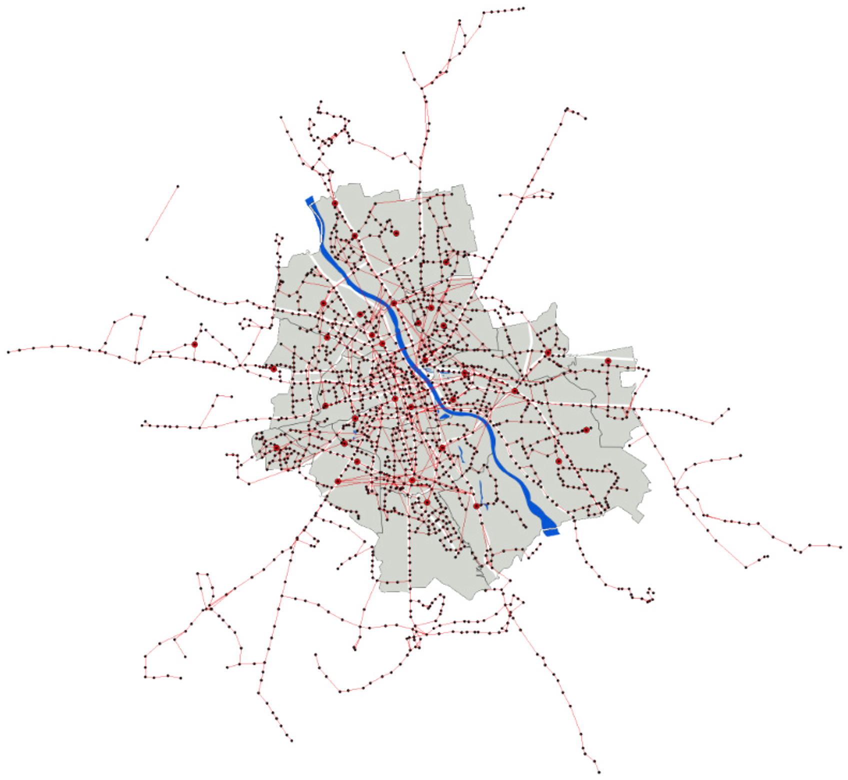

Figure 3: Result obtained by the MSBT algorithm - 35 points selected

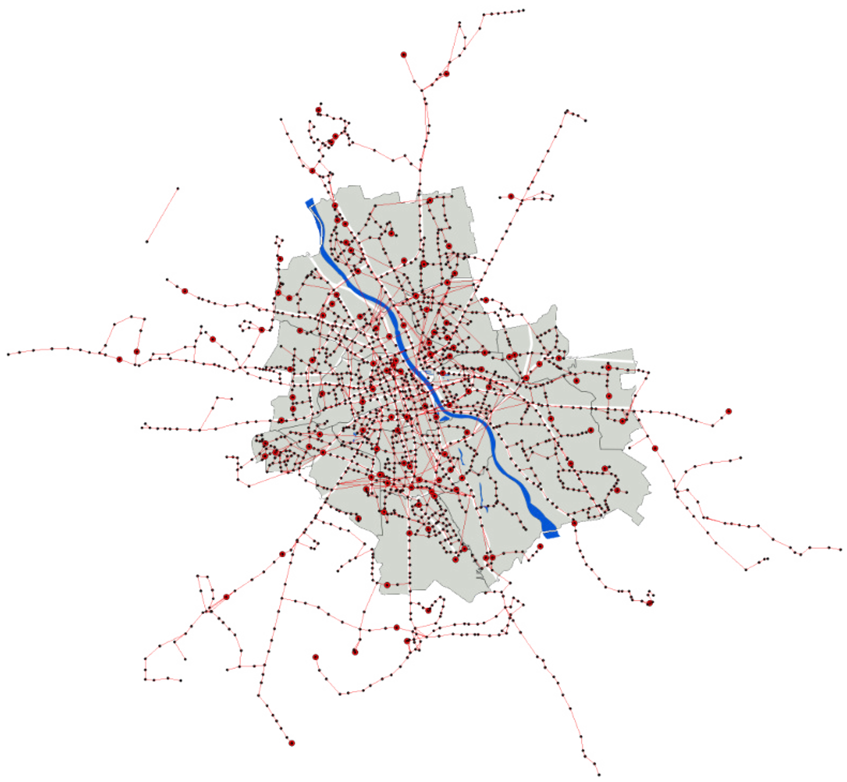

Figure 4: Result obtained by the RSBT algorithm - 140 points selected

VII. Conclusions

This article is devoted to the problem of locating hubs in a public transportation network. Two different approaches to this problem are presented. Namely, the first group of considered methods are based on greedy strategies, and the second group, using evolutionary strategy. In the computational experiments the model of the transportation network of Warsaw was analysed. The obtained results showed for this model the potential P&R set of 34 stations using the MSBT greedy method and of 140 potential points using the RSBT greedy method. Using the EA with imposed number of hubs, satisfactory solutions for 50, 100 and 152 predetermined numbers of candidates were obtained. Additionally, it must be concluded that the graph reduction could be used especially for the evolutionary approach as a filtration method significantly reducing the size of the solved problem. Almost always the preliminary reduction of the stop number speed up the computational process by about 46%. According to the quality of solutions obtained with and without the preliminary reduction it appears that the use of the proposed preprocessing method seems to be justified.

This work is the first step towards obtaining a more complete model of communication in Warsaw, which would be a base to find and project communication hubs with proper places for P&R facilities.

References

[1] Aros-Vera F., Marianov V., Mitchell J., p-Hub approach for the optimal park-and-ride facility location problem. European Journal of Operational Research 226, 2013,

[2] Bonato A., Lozier M., Mitsche D., Perez-Gimenez X., Prałat P., The domination number of on-line social networks and random geometric graphs, Theory and Applications of Models of Computation: 12th Annual Conference, Singapore, 2015

[3] Ding-Zhu Du, Peng-Jun Wan, Connected Dominating Set: Theory and Applications, Springer Optimization and Its Applications, 2012,

[4] Farhan B., Murray A. Siting Park-and-Ride facilities using a multi-objective spatial optimization model. Computers & Operations Research 35, 2008,

[5] Gartner B., Matousek J., Approximation Algorithms and Semidefinite Programming, Springer, 2012,

[6] Gogas P., Papadimitriou T., Matthaiou M., A Novel Banking Supervision Method using the Minimum Dominating Set, The Rimini Centre for Economic Analysis, 2014,

[7] Greetham D., Poghosyan A., Charlton N., Weighted -rate dominating sets in social networks, University of Reading, UK, 2014,

[8] Haynes T., Hedetniemi S., Slater P., Fundamentals of Domination in Graphs, Marcel Dekker Inc., 1998,

[9] Henning M., Yeo A., Total Domination in Graph, Springer Monographs in Mathematics, 2013,

[10] Jin-Hua Zhao, Habibulla Y., Hai-Jun Zhou, Statistical Mechanics of the Minimum Dominating Set Problem, Journal of Statistical Physics, 2015,

[11] Krishnam R., Indukuri R., Penumathsa S.V., Dominating Sets and Spanning Tree based Clustering Algorithms for Mobile Ad hoc Networks, International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February 2011,

[12] Mażbic-Kulma B., Sęp K., Some approximation algorithms for minimum vertex cover in a hypergraph. Computer Recognition Systems 2. Springer-Verlag, Berlin-Heidelberg, pp. 250-257. Series: Advances in Soft Computing, 2007,

[13] Milenkovic T., Memisevic V., Bonato A., Przulj N., Dominating Biological Networks, PLoS ONE, 2011,

[14] Molnár F. , Sreenivasan S., Szymanski B. K. & Korniss G. Minimum Dominating Sets in Scale-Free Network Ensembles, Nature, 2015,

[15] Naixue Xiong, Xingbo Huang, Hongju Cheng, and Zheng Wan, Energy-Efficient Algorithm for Broadcasting in Ad Hoc Wireless Sensor Networks, Sensors 2013,

[16] Nettleton D. F., Data mining of social networks represented as graphs, Computer Science Review, Elsevier, 2013,

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