DOI: 10.15439/2017F296
ISSN 2300-5963 ACSIS, Vol. 11

Decision Support System for Robust Urban Transport Management

Piotr Wiśniewski, Krzysztof Kluza and Antoni Ligęza
AGH University of Science and Technology
al. A. Mickiewicza 30, 30-059 Krakow, Poland
E-mail: {wpiotr, kluza, ligeza}@agh.edu.pl

***Abstract*—We present a decision support application which can be used for alternative route generation in case of tramway traffic disruptions. Our solution is based on a mixed graph network model, where vertices represent major points and edges are used to model track sections. The proposed application uses model data stored in a set of source files and enables the user to execute one of four algorithms which are useful for tramway traffic management in case of a crisis situation.**

***Index Terms*—decision support, graph theory, robust traffic management, route planing, public transport, crisis management**

I. Introduction

TRAFFIC congestion, especially in the major cities, is constantly growing. Simultaneously, the environmental awareness level is becoming higher and higher. Thus, these two factors cause the increasing role of the public transport in our everyday. Optimized transportation services are considered as a significant factor that makes the city more effective for its inhabitants as well as business entities [1], [2]. Such situation can be observed primarily in urban agglomerations with the advanced traffic systems and the variety of public transportation.

However, even in such developed areas, the urban transport management entities have to deal with some crisis situations. There are many random factors or infrastructure conditions which cause specific crisis issues.

From the traffic management perspective, the extreme cases require an intervention of a traffic controller like emergency rerouting of vehicles. Incorrect or lack of the decision can cause instability in the whole traffic system with the consequence of financial losses.

European Commission in [3] also takes into account the problem of public transport continuity assurance. One of their thought-provoking example concerning the disruption of transportation system was the example the eruption of Eyjafjallajökull volcano in Iceland in April 2010.

This work constitutes a continuation of our previous work [4], which introduced a mixed graph-based mathematical model for public transport networks. Here, we extend our previous contribution by describing additional route generation algorithms and presenting a sample application that enables their use in practice. We focus on using our solution in the area of tramway transit, as this means of transport is more exposed to crisis situations compared to road traffic.

The paper is organized as follows. Section II presents the existing works referring to public transport management. In Section III, a graph-based network model used in the proposed solution is presented. In Section IV, the algorithms for route generation are described. Application details are presented in Section V, and its usage examples are shown in Section VI.

II. Related works

There are several areas related to the subject our research, such as road traffic analysis, graph theory or vehicle route planning. As ad-hoc re-planning in case of crisis situation often leads to deterioration of plan quality, a robust route planning for passenger vehicles in city traffic was proposed by Ernst [5]. His approach, similar to our solution, allows for real-time robust route planning [6]. Mandziuk and Nejman [7] proposed a method for generating optimal routes for vehicle drivers who need to reach their customers using tree search algorithm. Another work presents a solution of a Capacited Vehicle Routing Problem using a Mixed Integer Linear Programming model [8]. Adamski [9], in turn, used stochastic processes for bus dispatching system. In [10], various types of decision support systems for vehicle fleet management were reported. In the context of a complex on-line control problem, one of such decision support system used in case of temporary railway track closures was presented in [11]. This approach combines the graphical power of Petri nets with the fuzzy sets which model rule-based expert system. Among the research concerned with optimal control of tramway networks, Blasum et al. [12] proposed three variants of solution for a problem of optimal tram scheduling in the morning – i.e. ordering tram assignments to departure. Winter and Zimmermann [13] extend this discussion on daily tram dispatching in localized depots.

III. Network model

The model used as a basis for the proposed application combines the approaches where directed [14] and undirected [15] graphs are used to represent public transport systems. In this paper, a mixed graph-based model was used, where directed edges correspond to tracks on double track sections and undirected edges are used to represent bi-directional single tracks. Its simplified form is presented by formula 1.

$$G = (V, E), \qquad (1)$$

where:

- V is a finite set of vertices,
- $E = E_1 \cup E_2$ is a set of both directed and undirected edges.

All the edges are assigned a vector weight function γ which has non-negative values of the planed time of ride and section length, expressed in minutes and kilometers, respectively. It is specified by formula 2.

$$\forall e \in E, \quad \gamma(e) = \{t, l\} \tag{2}$$

Vertices of the graph correspond to selected decision points in the network, such as: track junctions, waiting points before single tracks, terminuses, parking tracks, initial stops, entrances or exits of a depot. As most of those points have a limited space for cars, a non-negative vertex capacity function C. Its values are integers expressed in units of measure or number of cars. The capacity function is given by formula 3. In order to identify terminuses a logical function $\tau : V \longrightarrow \{0, 1\}$ was defined. Its value is equal to 1 if the selected vertex is a terminus and 0 otherwise.

$$C : V \longrightarrow \mathbb{Z}_{\geq 0} \tag{3}$$

In the first step of the model creation bi-directional track endpoints are connected with undirected edges while the other vertices are linked with directed edges regarding left-hand driving. An assumption is made that edges connecting two vertices within the area of one stop are given a zero-weight. The same rule applies to edges leading to single track sections.

In order to eliminate the error of changing car direction at an endpoint of a single track, the model definition contains a set of forbidden paths. A forbidden path is a path, whose any subsequence fulfills at least one of the following conditions:

- passing from one double-track section to another if there is an endpoint of a bi-directional track in between,
- entry to a single track section directly from an edge with a non-zero weight vector,
- if a vertex is an endpoint of an undirected edge and any other edge is chosen when generating path running through this vertex.

An example of a forbidden path is shown in Figure 1 which represents the "Ruda Połuniowa" passing loop, connecting two single track sections. According to this part of the network model there is a path designated by a sequence of vertices $(2, 3, 4)$. In real tramway networks following this route with a car having only one driver cabin is not possible, as it would need a change of direction, which in this case is allowed only on a loop or a turning triangle.

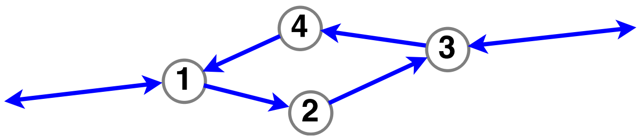


Figure 1. Model of the passing loop "Ruda Południowa".

A tramway line is a set of paths between two vertices marked as terminuses. As the route may differ depending on the direction and time of the day, the notion of variant is used to describe a specific run. Let $p(v_s, v_t)$ be a path from v_s to v_t, then a variant can be defined by formula 4.

$$w(v_s, v_t) = p(v_s, v_t) : \tau(v_s) = 1 \wedge \tau(v_t) = 1 \tag{4}$$

Therefore, a line is described as a set of variants marked by a specific number:

$$L(n) = \{w(v_s, v_t)\}, \quad n \in \mathbb{N}. \tag{5}$$

A train is a car or a set of cars in operation with respect to defined schedule or the orders of a traffic controller, running on a specific line or off-schedule. The solution proposed in this paper is applicable for planed trains which can be characterized by the following elements:

1) Train number (a line number concatenated with a running number in range 0-99).
2) Schedule, containing departure times from initial stops.

IV. Solving method

When there is a need to temporarily close a section of the track, which corresponds to a removal of an edge in the graph model, a train approaching the blocked section has to be redirected to an alternative route. Therefore it is needed to solve a path generation problem, by finding the best destination point and the shortest route connecting with that vertex the current position v_p which can be estimated based on the timetable or using a vehicle positioning system [16]. This route is calculated using the modified Dijkstra algorithm that excludes forbidden paths during the search by assigning infinite distances to vertices reachable by such a path.

A. *Route generation algorithm*

The algorithm used in the application presented in this paper consists in generating a shortest path to each reachable terminus. Routes leading to terminuses whose capacity is exceeded are excluded from the search. The alternative route is selected by maximizing a profit function. If p_0 is the subpath of the current variant, starting from vertex v_p then profit function Q is described by formula 6.

$$Q(p_0, p_a, p_c, v_a) = \frac{d(p_c) + \phi(v_a)}{|d(p_0) - d(p_a)|}, \tag{6}$$

where:

- p_a is the calculated alternative path,
- p_c is the common part of paths p_0 i p_a,
- v_a is the terminus of p_a,
- $d(p)$ is the length of path p,
- $\phi(v) = 1$ if v is the terminus of p_0 and 0 otherwise.

If two or more routes are assigned the same value of function Q the shortest one is chosen and if they have the same length, the one with a less occupied terminus is selected.

B. Time of ride calculation

In real transport systems the actual time of ride may differ from the scheduled one. Minor delays happen usually in case of high traffic congestion, passenger exchange as well as weather conditions. However, more significant disruptions may occur if a train runs off-schedule on a route containing single track sections. In such a case, when calculating the time of ride between two decision points, it is needed to take into account the time spent for waiting at the entrance of each single track. The algorithm used to calculate time of ride t_p through route p which contains single tracks is as follows:

1) Set $t_p = 0$ and t_0 as current time.
2) Calculate scheduled time t_1 to the nearest single track section. Set $t_p = t + t_1$.
3) Set t_2 as the scheduled time of ride through the single track section. If there is a train on this section in time interval $(t_0 + t_p, t_0 + t_p + t_2)$ then set $t_p = t_p + 1$ and re-execute this step. Otherwise set $t_p = t_p + t_2$.
4) If there are no more single track sections on the route then calculate scheduled time of ride t_3 until the end of the route and finish calculations setting $t_p = t_p + t_3$. Otherwise go back to step 2.

Using this method enables to estimate the real time of ride through a path, assuming that the train in the opposite direction runs without disruptions.

C. Return algorithm

After removing the effects of a crisis situation and re-opening the blocked section it is necessary to put the re-routed trains on their original paths and return to scheduled operation as soon as possible. A method that calculates the return scenario for train n_p and point v_s at time t_0:

1) Find the nearest scheduled departure of train n_p from initial stop v_p.
2) Generate the shortest path from point v_s to v_p and calculate time of ride t_p to this vertex.
3) If the arrival at point v_p occurs after the scheduled departure of the next train departing from this point, then set v_p equal to the initial stop of the next departure after time $t_0 + t_p$ and go back to point 2.
4) If the arrival at point v_p occurs after the scheduled departure of train n_p from this point or parking at point v_p is not possible (vertex capacity exceeded), then set $t_0 = t_0 + t_p$ and go back to point 2.
5) Finish calculations. Train n_p is back to schedule at time $t_0 + t_p$ at point v_p.

V. Application

The aim of the created application is to support dispatchers in the process of tramway traffic management in case of operation disruptions caused by crisis situations. Usually the decisions being taken in such cases are based on the network knowledge and work experience of the dispatcher. Unified criteria of alternative route generations were not formulated, which can lead to long lasting disturbances in urban transport circulation. The proposed system can be operated by one user, who works as a dispatcher and manages the tramway traffic within a designated area. People employed on this post are characterised by very good level of network knowledge, but in general they are not accustomed to use advanced computer tools. Moreover, there are often cases when disturbances occur in two or more places at the same time and that requires constant attention as well as contact with drivers and traffic control services. Therefore the decision support system for dispatchers needs to be simple and efficient, and provide a clear interface that demands from its user only data necessary for problem solving. The main requirement that was formulated for the application was the possibility to run it in various environments, without the necessity to use any specific equipment or dedicated software. Therefore source files should be saved in an open format which enables easy editing.

A. Usage scenarios

According to previous assumptions made in this section only one user is needed in the application and this person uses it on their workstation only. Therefore user authentication was excluded from the scope making the assumption that the user logs into their account in the operating system. The following usage scenarios were proposed for the application:

- display network information: decision points and section parameters,
- localize train on the route,
- generate alternative route for a train when a certain section is blocked,
- return to the schedule.

B. Data flow analysis

The input of the application is data inserted by the user and network model files. During the design phase a decision was made not to use an external database with complete network information. The reason for that decision was the simplification of the system. The assumption was made that the network model is uploaded to the system in form of source files during start-up. This solution is not optimal regarding operating memory usage, but enables the program to run without the necessity to maintain connection with a database server. A simplified data flow diagram [17], representing the proposed system's inputs and outputs was shown in Figure 2. Inbound data were marked with blue color while outbound data (information shown on the screen and logs saved to file) was marked with green color.

According to the diagram presented in Figure 2 inbound and outbound data are:

1) Selected function and, according to the scenario:
    a) section or decision point ID,
    b) train number and current time,
    c) train number, blocked section ID, current time,
    d) train number, train location, current time.
2) Model files: graph details, route list and schedules.
3) User information, according to the scenario:

a) section or decision point parameters,
b) train location and direction,
c) recommended alternative route and return path,
d) recommended direction and estimated time of the return to schedyle

4) Logs generated by the application and saved to a text file.

C. Proposed solution

A sample application was created in the widely used Java language, using the IntelliJ IDEA 14.1.1 and Java SDK 1.8 package. Graphical user interface was made using Java Swing library. The network was modelled using JUNG library (*Java Universal Network/Graph Framework*) [18], dedicated to represent both directed and undirected graph. The created model described in section III was implemented using *SparseGraph* which allows to perform operations on mixed graphs. This class enables the programmer to define vertices and edges as objects and contains methods that enable to:

- specify edge type (directed or undirected),
- determine vertices incident to the selected edge,
- determine the edge connecting two vertices,
- determine successors of a vertex.

Regarding the large number of parameters, presented in section III, vertices and edges of the network graph were defined as classes. Distances in kilometers are represented as decimals and time of ride is represented as integer and expressed in minutes, which is a dominant unit in public transport scheduling [19]. Additional classes were created also to represent trains and tramway lines. Variants of the latter, as well as other paths corresponding to tramway routes used in the system, are represented as lists of vertices IDs.

The length of tramcars used in Europe can be estimated between 10 and 55 meters [20], [21]. Therefore the capacity of vertices was expressed as integer in range 0-3, where one unit corresponds to 17 meters. A zero capacity was set in case when stoppage is not allowed at the selected point.

D. Model files

Information about the tramway network mathematical model are stored in four CSV files, each of which is used to create objects of the corresponding class. In order to use the application without errors it is necessary to define the following files:

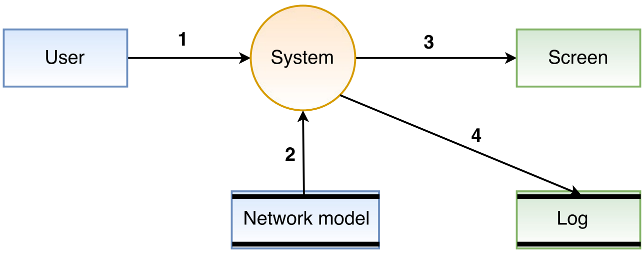
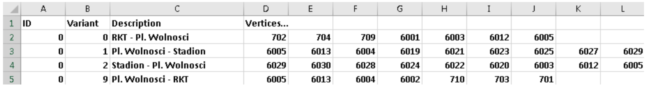


Figure 2. Simplified data flow diagram.

1) File *vertices.csv* which holds information about graph vertices. Columns: vertex ID, vertex name (8 characters), description, capacity, information, if the vertex is a terminus.
2) *edges.csv* stores information about graph edges. Columns: start vertex ID, end vertex ID, description, capacity, scheduled time of ride, length, information if the edge is undirected. During the upload of model files each edge is assigned a unique number, which is a concatenation of its endpoints IDs, and a 17-character name in form of its endpoints names connected by a dash. In case of an undirected edge the vertex with a lower ID is used at first.
3) *lines.csv* stores information about tram lines. Columns: line number, variant number, description, vertices on the route (one ID per column).
4) *tramcars.csv* stores information about trains and timetables. Adding a new train is executed by inserting the keyword *new* in a new row, along with the train number and vehicle type. The following vertices determine the timetable. Columns: departure time, variant number.

First row of every source file is a legend for used columns and is omitted during data upload. Part of the file *lines.csv* containing the definition of line "0" and its four variants is presented in Figure 3.



	A	B	C	D	E	F	G	H	I	J	K	L
1	ID	Variant	Description	Vertices...								
2	0	0	RKT - Pl. Wolnosci	702	704	709	6001	6003	6012	6005		
3	0	1	Pl. Wolnosci - Stadion	6005	6013	6004	6019	6021	6023	6025	6027	6029
4	0	2	Stadion - Pl. Wolnosci	6029	6030	6028	6024	6022	6020	6003	6012	6005
5	0	9	Pl. Wolnosci - RKT	6005	6013	6004	6002	710	703	701		

Figure 3. Source file with line definitions, opened in a spreadsheet editor.

VI. Usage examples

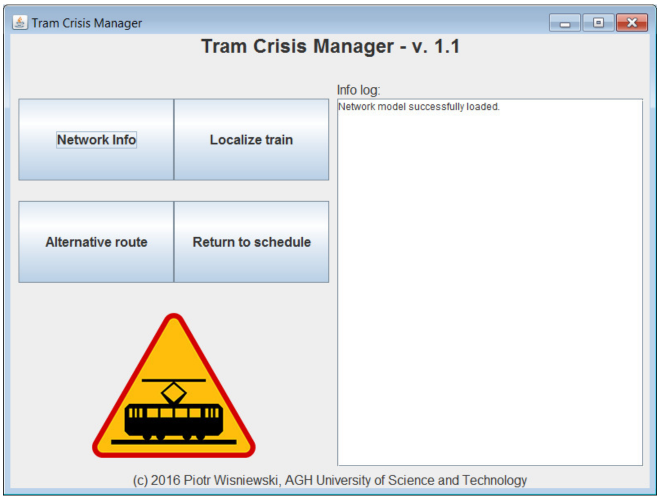
The application has a form of an executable JAR file which can be successfully run in Windows (7 or higher) as well es Linux (Ubuntu 14 or equal) environment. Application window is presented in Figure 4. Each of the usage scenarios is represented by a separate button.

Functionalities of the proposed application were tested using a model representing the central part of Upper-Silesian tramway network, which is the largest tramway system in Poland [22]. The test model is a mixed graph containing 135 vertices and 187 edges, 19 of which are undirected.

A. Network Information

Pressing the button *Network Information* executes the function which returns current information about the selected point or section. A pop-up window with input fields for current time and section/point ID appears on the screen. A sample result presenting section parameters and its occupancy was presented in Figure 6.

In *TramLogs.txt* file simplified logs, which contain information about the results of a query, are saved. Full point and section descriptions were replaced with their short names having 8 and 17 characters respectively (see section V-D for details).



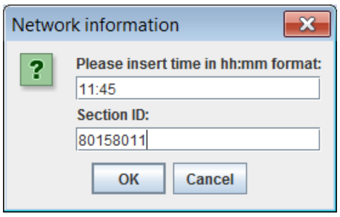
Tram Crisis Manager - v. 1.1

Info log:

Network model successfully loaded.

(c) 2016 Piotr Wisniewski, AGH University of Science and Technology

Figure 4. Initial window of the application run in Windows environment.



Network information

Please insert time in hh:mm format:
11:45

Section ID:
80158011

Figure 5. Network information data input.

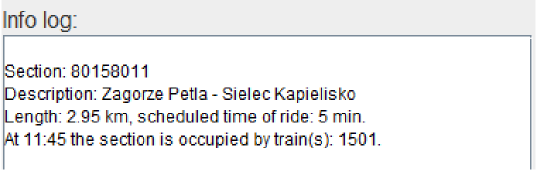
B. Train location

Train location is verified after pressing the appropriate button. The result is a point or section where the selected train is situated in the inserted time. In addition the user is informed about the line number and direction. In case of trains which are included in the list but are out of service, their depot is displayed as the current location.

C. Alternative route generation

The most important functionality of the application which is alternative route generation is executed by pressing the *Alternative Route* button. The user is asked for traffic disruption time and the train number, for which the route will be generated, as well as for ID of the blocked section. As a response the user is given a list of decision points for the generated route, its length and the estimated time of ride calculated regarding other vehicles moving on this route. In the next step a return path to the original route is recommended. Its departure time is equal to the arrival time on the selected terminus, if it is empty or to the departure of the last train that was stationing on this terminus.

A practical example of the application utility is as follows: it was admitted that on Saturday at 12:10 PM there was a collision of a tramway with a passenger car on a rail crossing in the city center of Katowice. The car is severely damaged and its immediate removal is not possible. In such case the dispatcher takes a decision to temporarily close the section shown in Figure 7. This section is included in the original route of line 0. Train 01, which is approaching the closed track, needs to be directed to an alternative route. After inserting data to the application the user gets the response shown in Figure 8.



Info log:

Section: 80158011
Description: Zagorze Petla - Sielec Kapielisko
Length: 2.95 km, scheduled time of ride: 5 min.
At 11:45 the section is occupied by train(s): 1501.

Figure 6. System response with network information.

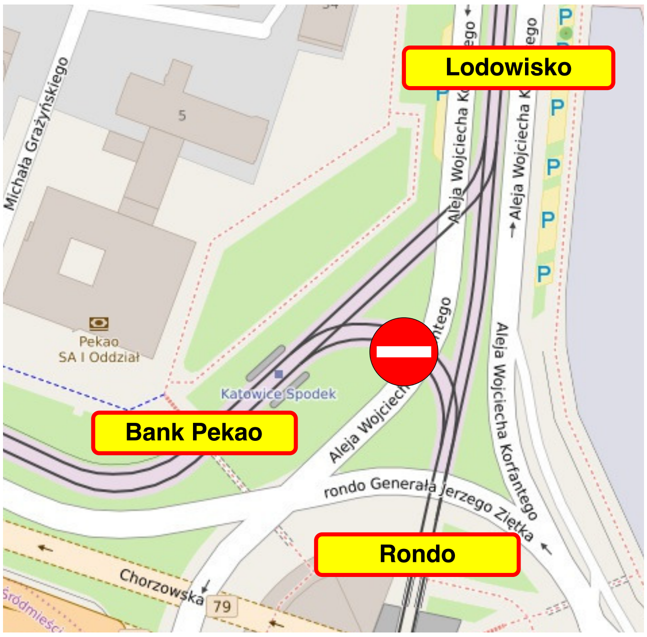
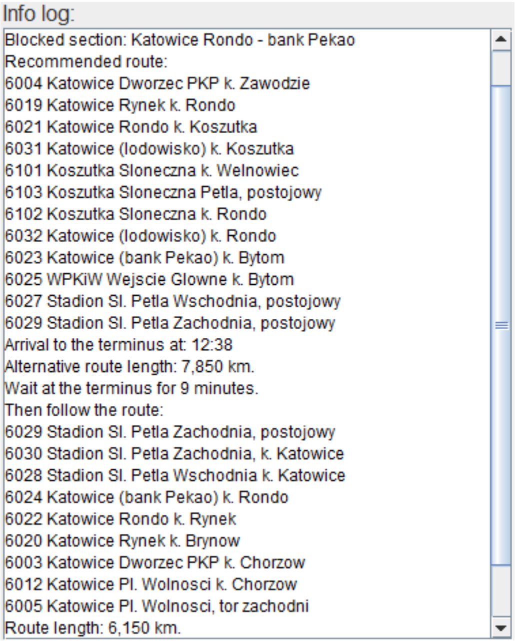


Figure 7. Closed section as a result of a collision [own work based on openstreetmap.org].

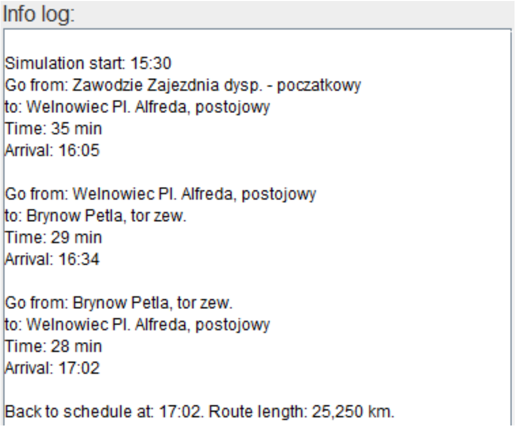
D. Return to schedule

After re-opening the blocked section it is necessary to put the trains back on their original routes. The user executes the appropriate procedure by pressing the button *Return to schedule*. In the next step the number of the train, which is off-schedule, as well as its location (nearest decision point) and the starting time are inserted. This functionality can be illustrated by the following example: as a result of the overhead line failure the power was turned off at 3 PM on the section "Rynek - Rondo" in the city center of Katowice. Train 161 was directed to an alternative route towards Zawodzie depot. Half an hour later, at 3:30 PM, the line was repaired and the dispatcher re-opened the section. In such a case all the trains operating off-schedule have to come back to their original routes as soon as possible. Therefore the dispatcher runs the procedure whose results are presented in Figure 9.



```
Blocked section: Katowice Rondo - bank Pekao
Recommended route:
6004 Katowice Dworzec PKP k. Zawodzie
6019 Katowice Rynek k. Rondo
6021 Katowice Rondo k. Koszutka
6031 Katowice (lodowisko) k. Koszutka
6101 Koszutka Sloneczna k. Welnowiec
6103 Koszutka Sloneczna Petla, postojowy
6102 Koszutka Sloneczna k. Rondo
6032 Katowice (lodowisko) k. Rondo
6023 Katowice (bank Pekao) k. Bytom
6025 WPKiW Wejscie Glowne k. Bytom
6027 Stadion Sl. Petla Wschodnia, postojowy
6029 Stadion Sl. Petla Zachodnia, postojowy
Arrival to the terminus at: 12:38
Alternative route length: 7,850 km.
Wait at the terminus for 9 minutes.
Then follow the route:
6029 Stadion Sl. Petla Zachodnia, postojowy
6030 Stadion Sl. Petla Zachodnia, k. Katowice
6028 Stadion Sl. Petla Wschodnia k. Katowice
6024 Katowice (bank Pekao) k. Rondo
6022 Katowice Rondo k. Rynek
6020 Katowice Rynek k. Brynow
6003 Katowice Dworzec PKP k. Chorzow
6012 Katowice Pl. Wolnosci k. Chorzow
6005 Katowice Pl. Wolnosci, tor zachodni
Route length: 6,150 km.
```

Figure 8. Alternative route generated for train 01



```
Simulation start: 15:30
Go from: Zawodzie Zajezdnia dysp. - poczatkowy
to: Welnowiec Pl. Alfreda, postojowy
Time: 35 min
Arrival: 16:05

Go from: Welnowiec Pl. Alfreda, postojowy
to: Brynow Petla, tor zew.
Time: 29 min
Arrival: 16:34

Go from: Brynow Petla, tor zew.
to: Welnowiec Pl. Alfreda, postojowy
Time: 28 min
Arrival: 17:02

Back to schedule at: 17:02. Route length: 25,250 km.
```

Figure 9. Recommended return plan for train 161.

VII. Conclusion

In this paper, we presented a decision support system for robust traffic management. Using the proposed application is possible in different software environments, without the necessity of using additional equipment. Its practical use may lead to more efficient reactions for crisis situations occurring in public transport. A significant feature of our approach is the mixed graph-based network model which can be used to represent various types of tramway networks, including bi-directional single track routes. In our future research we plan to extend the algorithm by allowing it to generate alternative routes for more trains at one time and provide complex information for the user as well as improving a data structure containing information about the model and linking it to the existing tramway databases.

References

[1] B. Gontar, Z. Gontar, and A. Pamuła, "Deployment of smart city concept in Poland. Selected aspects." *Management of Organizations: Systematic Research*, no. 67, pp. 39–51, 2013.

[2] R. Klimek and L. Kotulski, "Towards a better understanding and behavior recognition of inhabitants in smart cities. a public transport case," in *Proceedings of 14th International Conference on Arficial Inteligence and Soft Computing (ICAISC 2015), 14–18 June, 2015, Zakopane, Poland*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 9120. Springer Verlag, 2015, pp. 237–246.

[3] E. Comission, "Continuity of passenger mobility following disruption of the transport system," Brussels, 2014.

[4] P. Wiśniewski and A. Ligęza, "An approach to robust urban transport management. mixed graph-based model for decision support," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2017, pp. 347–356.

[5] S. Ernst and A. Ligęza, "A rule-based approach to robust granular planning," in *International Multiconference on Computer Science and Information Technology*, Wisła, 2008, pp. 105–111.

[6] S. Ernst, "Artificial intelligence techniques in real-time robust route planning," Ph.D. dissertation, AGH, Kraków, 2009.

[7] J. Mandziuk and C. Nejman, "Uct-based approach to capacitated vehicle routing problem," in *Artificial Intelligence and Soft Computing: 14th International Conference, ICAISC 2015*, Zakopane, 2015, pp. 679–690.

[8] P. Sitek and J. Wikarek, "A hybrid method for modeling and solving constrained search problems," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds. IEEE, 2013, pp. 385–392.

[9] A. Adamski, "Discon: Public transport dispatching robust control," in *EWGT2013 – 16th Meeting of the EURO Working Group on Transportation*, Porto, 2014, pp. 1206–1216.

[10] J. Żak, "Decision support systems in transportation," in *Handbook on Decision Making*. Berlin Heidelberg: Springer-Verlag, 2010, pp. 249–294.

[11] A. Fay, "A fuzzy petri net approach to decision-making in case of railway track closures," in *IFSA World Congress and 20th NAFIPS International Conference, 2001.*, 2001.

[12] U. Blasum, M. R. Bussieck, W. Hochstättler, C. Moll, H.-H. Scheel, and T. Winter, "Scheduling trams in the morning," *Mathematical Methods of Operations Research*, vol. 49, no. 1, pp. 137–148, 1999.

[13] T. Winter and U. Zimmermann, "Real-time dispatch of trams in storage yards," *Annals of Operations Research*, vol. 96, no. 1, pp. 287–315, 2000.

[14] D. Lückerath, O. Ullrich, and E. Speckenmeyer, "Modeling time table based tram traffic," *Simulation Notes Europe*, vol. 22, no. 2, pp. 61–68, 2012.

[15] T. Schlechte, "Railway track allocation: Models and algorithms," Ph.D. dissertation, Technische Universität Berlin, Berlin, 2012.

[16] C. Sungur, H. B. Gökgündüz, and A. A. Altun, "Road vehicles identification and positioning system," in *2014 Federated Conference on Computer Science and Information Systems*, 2014, pp. 1353–1359.

[17] P. Szwed, "Metodologia SART Warda-Mellora," [online], http://home.agh.edu.pl/~pszwed/se/sart/kss09.html. Dostęp: 2016-08-12.

[18] "JUNG - Java Universal Network/Graph Framework," [online], http://jung.sourceforge.net/. Accessed: 2016-08-12.

[19] C. Liebchen, "Periodic timetable optimization in public transport," in *Operations Research Proceedings 2006: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Karlsruhe, September 6–8, 2006*, K.-H. Waldmann and U. M. Stocker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 29–36.

[20] A. Lubka and M. Stiasny, *Atlas Tramwajów*. Kolpress, 2011.

[21] "Longest tram enters service in Budapest," [online], 2016, http://www.railwaygazette.com/news/single-view/view/longest-tram-enters-service-in-budapest.html. Accessed: 2017-05-07.

[22] J. Drogoś, "Charakterystyka sieci tramwajowej górnośląskiego okręgu przemysłowego," in *Tramwaje w Polsce*. Łódź: Księży Młyn, 2013, pp. 74–87.