HEURISTIC BASED REAL-TIME TRAIN RESCHEDULING SYSTEM

Snezana Mladenovic[1], Slavko Veskovic[2], Irina Branovic[3], Sladjana Jankovic[4], Slavisa Acimovic[5]

[1,2,4,5] Faculty of Transport and Traffic Engineering, University of Belgrade
[3] Mathematical Institute of the Serbian Academy of Sciences

Abstract

This paper deals with the problem of train rescheduling on a railway network. Starting from a defined network topology and initial timetable, the paper considers a dynamic train rescheduling in response to disturbances that have occurred. The train rescheduling problem was mapped into a special case of the job shop scheduling problem and solved by applying a constraint programming approach. To improve the time performance of available constraint programming tool and to satisfy a selected objective function, a combination of three classes of heuristics are proposed: bound heuristics, separation heuristics, and search heuristics. Experimental evaluation of the implemented software in Belgrade railway dispatching area indicates that the proposed approach is capable of providing the support to a real-life operational railway control. In our solution, the dispatcher has the possibility of choosing the most suitable optimization criterion from the set of seven available ones.

Keywords: Railway, Train rescheduling, Job Shop Scheduling, Constraint Programming, Heuristics

1 Introduction

Operational control in railway traffic is supposed to amend the current timetable, taking into consideration the complex dynamic interactions among the events, objective function, and evaluation of the future system state. The initial train timetable must sometimes be modified because of accidents, natural disasters, engine troubles, human factors; this task is called train rescheduling [10].

The problem of train scheduling is known to be NP-hard, i.e. an algorithm for solving this problem in polynomial time is believed to be non-existent [6]. The train scheduling problem considered for a larger fragment of railway network, a longer planning period, or higher number of trains, is a part of designing the timetable carried out at the level of tactical planning. The train timetabling problem has been widely studied in the literature. This refers to [4], [8] and [19] for surveys of the literature on timetable planning. In [4] there is an excellent overview of the state-of-the-art on train timetabling, underlining the differences between models and methods that have been developed to tackle the nominal and the robust (fault-tolerant) versions of the problem. There are two main variants of the nominal problem: periodic (cyclic, where schedule repeats every given time period, for example every hour) and non-periodic (non-cyclic). Passenger traffic is almost always performed according to the periodic timetable [8]. Thus, [18] presents a periodic timetable which has been implemented for Berlin subway using integer programming formulation of the periodic event scheduling problem (PESP). Flexible PESP formulation is used in [7] for the generation of flexible time slots for the departure and arrival times of trains instead of the exact times. That article focuses on the periodic timetable on the macroscopic level which is the input for the microscopic level, i.e. for finding a feasible schedule. In circumstances when a large number of routes is on-demand and when there are several train operators using the same infrastructure, the non-cyclic timetable is more appropriate [3].

Train rescheduling, e.g. operational reconstruction of the timetable, is subsequently performed on a smaller fragment of railway network, over a shorter planning period, in response to disturbances that have arisen. The rescheduling may be considered a more difficult problem compared to initial scheduling. According to [27], major reasons for this are as follows:

- it is difficult to choose an objective, uniformly applicable criterion for rescheduling;
- train rescheduling is a large size combinatorial problem;

- high immediacy is required;
- necessary information cannot be always obtained.

In the past, human experts, called train dispatchers, accomplished rescheduling process using simple information available at the time of dispatching. This approach was not satisfactory, since it solved the problem only locally. Early rescheduling systems mostly tested if the solution proposed by the user was feasible at all, whereas avoiding full schedule regeneration [9]. Nowadays, software systems based on fast algorithms help dispatchers to quickly produce a "good enough" rescheduling plan.

The basic idea behind our research was to create the most realistic model of a railway network and to develop original heuristic algorithms, which in conjunction with constraint programming mechanisms become capable of finding a "good enough" solution to the train rescheduling problem within the limited time. As opposed to dispatching rules permitting quick decision-making, but not taking into account global information, the constraint programming approach carries out a systematic search and as such, it is time-consuming, but also capable of finding better solutions, in concordance with the assigned objective function. To improve the time performance of available constraint programming tool, we proposed three classes of mutually interacting heuristics, called bound heuristics, separation heuristics, and search heuristics.

The rest of the paper is arranged as follows: Section 2 describes the selected previous work in train rescheduling. Section 3 defines the railway network topology in a concise and precise way. Section 4 maps the problem of train scheduling on a single-track line to the job shop scheduling problem with additional constraints. The fifth Section describes the solution to the problem by using constraint programming approach. After the train scheduling problem has been defined as a Constraint Satisfaction Optimization problem, the heuristics for a faster solution are discussed. In the sixth Section, the proposed method is evaluated on a real example of Belgrade railway node. The final considerations and possible lines of further research are presented in the last Section.

2 Related work

The traditional survey of [10] reviews a large number of papers dealing with train routing, train scheduling, and real-time traffic management.

A newer excellent survey of approaches for railway traffic scheduling and rescheduling can be found in [28]. The related approaches are classified based on whether they have been tested on fictive or real data, what kind of railway infrastructure the models are able to represent, whether they consider a single line or whether they can represent a network. Regarding the objectives applied, most often the objective function minimizes the total delay and the weighted delays. Rescheduling approaches mostly focus on a railway system with either passenger or freight traffic, and traffic on a single-track line rather than a network and heterogeneous traffic.

The latest, very extensive survey of models and algorithms for real-time railway rescheduling is presented in [5] and appeared in the final phase of presenting this research. Authors give an overview of a number of papers discussing three types of real-time railway rescheduling procedures: the timetable, the rolling stock, and the crews. Of special interest to our work are papers dealing with real-time timetable rescheduling. The authors of this survey initially classify these papers according to whether they solve the problem disturbances (relatively small perturbations that influence the railway system) or disruptions (relatively large external incidents that lead to the cancellation of a number of trips in the timetable). Considering a definition given in this work, which describes a disturbance as such that it can be handled by rescheduling the timetable only, without rescheduling the resource duties, our research solves the problem disturbances. Our train rescheduling consists of delaying the departure and arrival times of the trains, and as a consequence, of possible modification of their order in the stations. Another classification presented by the authors of the survey concerns the level of detail considered in the railway system. The macroscopic level considers the railway network at a higher level, when the details of block sections and signals are not taken into account. In a microscopic approach, these aspects are considered in detail. We consider the actual line-side signals that limit resources, and for this reason, our approach is microscopic. And finally, the research may be oriented to the trains (delayed or canceled) or to passengers/freight.

In view of this extensive survey, we directed our review to more recent papers or works that are closest to our research.

The daily real-time problem of dispatchers is known as conflict detection and resolution problem and it was solved by [12]. Their paper presents a real-time traffic management system, called ROMA (Railway traffic Optimization by Means of Alternative graphs). ROMA is able to optimize the railway traffic also when the timetable is not conflict-free. This fact enables its usage when managing railway traffic in case of severe traffic

disturbances, such as when emergency timetables are required and traffic dispatchers need support to solve conflicts.

Authors in [27] used passenger dissatisfaction as the objective criteria of train rescheduling problems and introduced an algorithm combining PERT (Program Evaluation and Review Technique) and simulated annealing. Their algorithm allows train cancellation, change of train-set operation schedule, change of track in a station etc. However, their system takes into account only passenger trains.

Study [14] was aimed at creating a relaxation adjustment plan first, and then choosing the earliest conflict to be solved according to priority. Operating plan was obtained after all the conflicts in the relaxation plan were eliminated in sequence.

Paper [22] describes the practical implementation of a real-time traffic management system whose aim is to test the feasibility of a completely automated system for conflict resolution and speed regulation.

Focus in [26] is on the resolution of train conflicts and it proposes a constraint programming formulation for the compound routing and sequencing problem. Computational experiments show that a truncated branch and bound algorithm can find satisfactory solutions for a rail junction within three minutes of computation time. Compared to the effects of decisions applied by the French operator SNCF for the railway network around the Pierrefitte-Gonesse junction north of Paris, the constraint programming techniques are able to reduce the delays by 62-95%. Work [29] presents a model for rescheduling trains in a rail network with several merging and crossing points. This paper considers the problem of rescheduling railway traffic on the southern part of the Swedish railway network connecting Stockholm, Gothenburg, and Copenhagen (in Denmark). The model contains continuous variables for representing start and end times of an event, and continuous variables that represent the delay of an event. In addition, it contains binary variables to express whether an event uses a track, and binary variables are used to decide the order of trains. The model considers the fixed headway times between trains and fixed running times along segments between stations. The problem is formulated as a mixed-integer linear program and solved by applying commercial constraint programming tools. The goal is to minimize a cost function based on the train delays. Four strategies are proposed for reducing the solution space based on restrictions on reordering and rerouting actions.

The authors of [1] used a similar mixed-integer programming formulation with two main differences. First, they considered explicitly the fact that unplanned stops will change the minimum and maximum allowed travel times because of acceleration and braking. The second difference was the modification of some constraints to admit more than one train on open-line section running in the same direction. This approach to train sequencing is very close to our research. The problem is solved by limiting the search space around the original non-disrupted timetable. The authors concluded that the average computation time less than 5 minutes is viable in practice in a real-time rescheduling context.

In the available literature, paper [17] deals with approximate time headway and not with real spatial headway of trains, which may be notably different if there is a significant difference in the length of resources and in the running speeds of trains. Therefore, these authors simplify the network infrastructure by assuming that the whole open-line section between stations is one block section, which does not correspond to a real-life case. For this reason they are forced to use approximate time headway instead of real spatial headway between trains that we used in this research. Also, as opposed to [24] the traffic mixture has been taken into account in this paper and our previous work [23], so that different priorities are assigned to different train categories. These authors study a simple single-track network, without junctions either in stations or on open-line sections between stations, which is the main difference from our research. The work presented here can be considered as an extension of our previous research described in [23], where the focus was on describing heuristics, and the implemented software system was at the first prototype level. In this work, we updated the software to the complete user-friendly decision-support system called TimeRec. The concept of surrogate objective function has not been implemented in TimeRec rescheduling system because experiments have proven that in some rare cases the surrogate objective function generates less acceptable schedule respect to the one generated by a dispatcher, for which the dispatcher may be reluctant to use the software at all.

A stream of research very closely related to ours is presented in [11], which analyzes the bi-objective problem of minimizing train delays and missed connections in order to provide a set of feasible non-dominated schedules. Also, paper [13] describes a dispatching system to pro-actively detect and globally solve the conflicts and to predict railway traffic based on the actual track occupation, signaling system and dynamic train characteristics. They used an alternative graph model.

Finally, [16] modeled the railway network in a similar way as we do: the movement of a train on the network is controlled by signals which divide the network into track sections called blocks. They formulate the problem of assigning trains to blocks as they pass through a network after a disturbance as a job shop scheduling problem with additional constraints. Three different types of disturbances are considered: small (shorter than 15 minutes), medium (between 15 and 30 minutes), and large (more than 30 minutes). However, as opposed

to our approach, they have a unique objective: to minimize the total weighted tardiness. The authors propose three modified versions of the shifting bottleneck procedure for solving the job shop scheduling problems. Tests have been performed on real-life instances from the very dense and congested London Bridge area in the United Kingdom, where the partial network is about 15 km long with a few busy stations. The computation times of 26 minutes are not realistic in terms of real-time decision making, however authors are planning to improve the solution time and the quality of the algorithm with more efficient heuristics that can be embedded in the current framework and exploit potential computational speedups.

3 Railway network topology

The railway network elements are integral parts of lines and stations – facilities, resources; we shall denote them as set R. According to the properties concerning the possible numbers of simultaneously present trains on a facility (i.e. its capacity), the numbers of entry and exit points of the facility and the possibility of connection, three disjunctive subsets can be distinguished: block sections – set P ($Type(r) = bs$, $Capacity(r) = 1$, if $r \in P$), entry-exit facilities – set U ($Type(r) = ee$, $Capacity(r) = 1$, if $r \in U$), and station tracks – set S ($Type(r) = st$, $Capacity(r) > 1$, if $r \in S$). Facilities from set P have exactly one entry and exit point; facilities from set S have mutually equaled and still higher than one, the numbers of entry and exit points; facilities from set U have different number of entry and exit points. The entry and exit points of the facility are provided with signals controlling its occupation. The railway network is built by connecting the exit points of one facility to the entry points of the other facility.

Open-line sections between stations consist of one or more block sections (*bs* resources) between which junction points may be found (*ee* resources). This makes it possible that a number of trains moving in the same direction may be present simultaneously on the open-line section between stations, where the distance between them is real, spatial. The increase of the number of facilities certainly makes the train scheduling problem more complex.

The example of a railway network built according to the described rules is presented in Figure 1 (part of Belgrade railway node). Resources of the railway network within number range from 1 to 16: $P = \{4, 5, 6, 10, 11, 14\}$, $U = \{3, 7, 9, 12, 15\}$, $S = \{1, 2, 8, 13, 16\}$ and ($Capacity(r) = 2$, if $r \in S$). There are two single-track sections between stations, of which one has a junction point (resource no. 12).

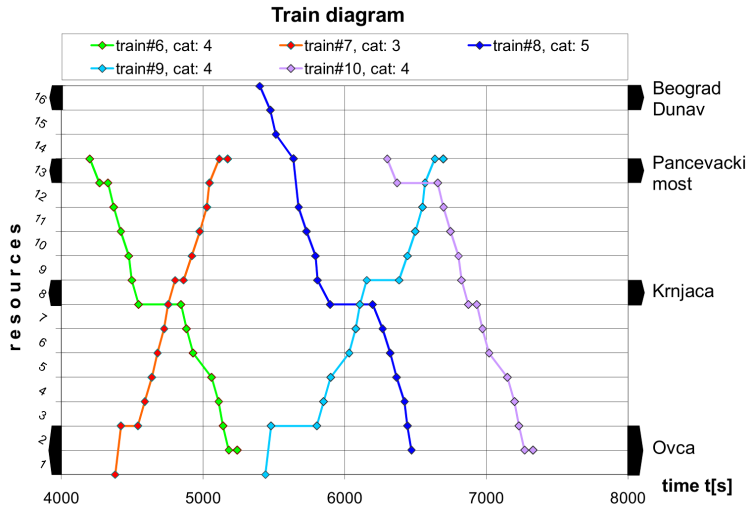

FIG.1. A real example of the railway network.

On railway lines with two-way traffic the train movement directions are traditionally designated as *odd* and *even*.

4 Mapping the single-track train scheduling problem into the job shop scheduling problem

In real service and production systems, there is often need to make decisions concerning the allocation of a limited number of types and limited quantities of different *resources by jobs* over time, along with the optimization of some objective function. Each job consists of one or more stages requiring time and resources and they are referred to as *activities*, *operations,* or *tasks*.

The scheduling model of interest for the presentation in this paper is the dynamic job shop problem [2, 25]. The model is a complex processing system with several resources and several activities, where each job has its inherent sequence of operations and inherent arrival time.

The constraints regarding the fixed order of operations for each of the jobs are referred to as *conjunctive constraints*. The constraints related to jobs processed on the same machine are generally called *disjunctive constraints*. It is possible to seek the solution to the job shop problem under *special constraints*.

The *feasible schedule* is any one that satisfies all imposed constraints and finds an actual start time for each operation on each of the jobs.

The initial timetable is an entry into the operational railway control. The timetable specifies the starting and final points of the journey as well as the scheduled arrival and departure times for each intermediate station en route. The timetable is said to schedule trains on a given railway infrastructure.

A real route is a series of all stations through which a train must pass from the origin to the destination. This paper interprets the term route in a somewhat modified way. The route is a *sequence of facilities* the train must cross on its journey from the origin to the destination. Instead of the arrival and departure times for each train and each facility on its route, we will assume that we know the ideal duration of occupation of each facility by train on its route. This occupation includes both the movement and any planned stopping. Since we know the planned time of train arrival to the first facility on the route, i.e. *planned train generation*, based on an ideal duration of occupation, we can assume that the *ideal train timetable* is known.

Each train is accompanied by a trip in a unique way. The train movements can also be considered as jobs to be scheduled to the infrastructure elements – resources. Thus established correspondence entitles us not to make a strict distinction among "train", "trip" and "job" in this paper.

The conflicts among trains arise when a number of requests for a resource exceed its capacity, or when some of the imposed constraints have been disturbed regarding the train movement control. In the general case the solving of conflicts requires an introduction of delay into at least one of the conflict trips.

4.1 Timetable scheduling in job shop terms

The mapping of a train scheduling problem into a job shop problem is an already known approach ([20, 24, 26]). However, the authors of the previous works mostly apply approach that differs from ours to a certain extent. Paper [24] models the stations as infinite capacity resources, which differs significantly from our realistic approach. Article [20] presents a model of train movement through metro station as a no-wait job-shop scheduling problem. Considering that this paper deals with the problem of routing and scheduling trains in the station, the authors model the station in detail as a set of track segments. A stopping point is a track segment in which a train can stop to execute a service. An interlocking route is the rail track between two stopping points, and it is actually formed by a sequence of track segments. All resources are single-capacity, as opposed to our model where the capacity of the station tracks is larger than 1, as defined earlier in Section 3. The model described in work [24] is unacceptable in our case because it forbids waiting for resources, i.e., prolonged dwelling in the station. Allowing prolonged dwelling is necessary in rescheduling circumstances when overtaking and crossing stations are determined dynamically. In our research, the routes of trains are fixed as opposed to research described in [26] where the routes are alternative.

Our mapping of the train scheduling problem into a special case of the job shop scheduling problem has been made as follows:

Let $R = P \cup U \cup S = \{r_1, r_2, ..., r_m\}$ be a set of railway infrastructure facilities available, $J = \{J_1, J_2, ..., J_n\}$ set of jobs, and N is the railway network. Each train trip $J_i \in J$ is a series of k_i operations $J_i = (o_{i_1}, \ldots, o_{i_{k_i}})$. To each train trip J_i there is a corresponding "one to one" mapping $\phi_i : \{1, 2, ..., k_i\} \rightarrow R$, which allocates to each operation o_{ij} of this trip a facility $\phi_i(j)$ from R on which this operation is planned to be processed. $\phi_i(1), \phi_i(2), \ldots, \phi_i(k_i)$ must represent one path on the network (when *direction*(J_i) is odd) or its inversion, (*direction*(J_i) is even). Each operation o_{ij} has a fixed processing time p_{ij} corresponding to the time of facility occupation $\phi_i(j)$ by job J_i. Function w joins to each job J_i its priority w_i. Each train is assigned its category cat_i, for which we shall assume to specify all train attributes.

The planned start time d_{ij} for each operation o_{ij} equals the earliest possible time, i.e. the earliest completion

TABLE 1. A train timetable for railway network from Figure 1.

Job≡ Train	Resource order ≡Train trip	Job gener. time d_i[s]	Operation processing time p_{ij}[s]	Categ. cat_i	Priority w_i
J_1	1,3,4,5,6,7,8,9, 10,11,12,14,15,16	0	28, 21, 56, 46, 54, 21,30, 29, 63, 57, 36, 125, 38, 75	5	1
J_2	13,12,11,10,9, 8,7,6,5,4,3,2	60	129, 40, 50, 56, 22, 108,39, 47, 130, 50, 31, 100	4	2
J_3	13,12,11,10,9, 8,7,6,5,4,3,2	300	45, 28, 50, 56, 13, 26, 9, 47, 40, 50, 18, 24	1	4
J_4	1,3,4,5,6,7,8,9, 10,11,12,13	1200	100, 48, 50, 130, 47, 29,108, 59, 56, 50, 18, 129	4	2
J_5	13,12,11,10,9, 8,7,6,5,4,3,2	2040	45, 28, 50, 56, 13, 26, 9, 47, 40, 50, 18, 24	1	4
J_6	16,15,14,12,11, 10,9,8,7,6,5,4,3,2	3000	75, 38, 125, 36, 57, 63, 14, 30, 10, 54, 45, 56, 21, 28	5	1
J_7	1,3,4,5,6,7,8, 9,10,11,12,13	3300	100, 48, 50, 40, 47, 29, 108,59, 56, 50, 18, 129	3	2
J_8	13,12,11,10, 9,8,7,6,5,4,3,2	5100	45, 28, 50, 56, 13, 26, 9,47, 40, 50, 18, 24	1	4
J_9	1,3,4,5,6,7,8, 9,10,11,12,13	5340	100, 48, 50, 130, 47, 29, 108, 59, 56, 50, 18, 129	4	2
J_{10}	13,12,11,10,9, 8,7,6,5,4,3,2	5640	129, 40, 50, 56, 22, 108,39, 47, 130, 50, 31, 100	4	2

time of the preceding operation: $d_{ij} = d_{i(j-1)} + p_{i(j-1)}$, $2 \leq j \leq k_i$, and $d_{i1} = d_i$, where d_i is the planned job generation time of J_i. If c_i denotes the planned, and C_i the actual job completion time, then the tardiness T_i of the job is defined as $T_i = \max(C_i - c_i, 0)$. We shall assume that the planned job completion time is the earliest possible, i.e. $c_i = d_i + \sum_{j=1}^{k_i} p_{ij}$.

The problem of determining the timetable, i.e. train scheduling over time consists of finding actual start time $\bar{d_{ij}}$, $\bar{d_{ij}} \geq d_{ij}$, for each operation on each of the jobs avoiding conflicts, meeting additional constraints and optimizing the selected objective function. If the ideal timetable is feasible, $\bar{d_{ij}} = d_{ij}$ will apply to all operations on each of the jobs.

Consider the railway network presented in Figure 1 with actual train categories operating on it: international long distance trains (category 1 - priority 4, the highest), intercity trains (category 3 - priority 2), city trains (category 4 - priority 2), and freight trains (category 5 - priority 1, the lowest). Priorities are assigned to trains by an expert. A timetable for this part of network is shown in Table 1. Even in case when no special constraints exist, the ideal timetable is not feasible, because it has six conflicts as consequence of disjunctive constraints. The visualization of conflicts is given in Figure 2 in the form known in railway traffic as the train diagram. It is actually a modified Gantt chart constructed based on tabular data.

4.2 Problem constraints

The constraint component in our research consists of four classes of constraints.

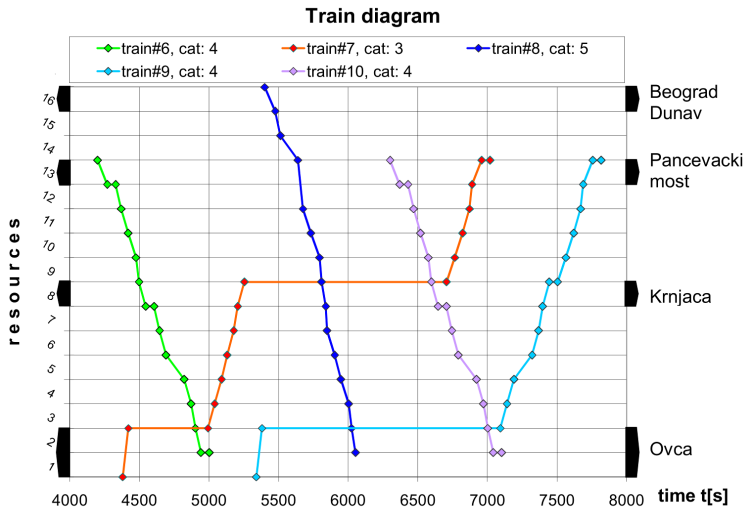

FIG.2. An infeasible timetable with six disjunctive conflicts on single-track sections.

1. Since the train scheduling problem is formulated as the job shop scheduling problem, it is clear that the fixed route corresponds to conjunctive constraints, and the constraints related to job processing on the same resource, taking into consideration its capacity, are normal disjunctive constraints.

2. The following set of constraints is related to preventing train collisions and it exists on each railway system. In the research that has been carried out, a minimum set of eight safety constraints has been defined, covering all known regulations applicable in real-time train movement on a single-track line on the Belgrade railway node. These constraints are:

Rule on speed

Train speed of a resource is determined by resource properties and train properties, i.e., direction of its movement and its category. It is assumed that the speed is constant and equals the minimum of maximum speeds allowed for certain resource and certain trains. Therefore, if for a train J_i $\phi(j) = r_l \in R$, then the maximum allowed speed V_{ij} of train J_i on resource r_l (where j-th operation o_{ij} will be done) equals $V_{ij} = min\{V_{max}(r_l, direction(J_i)), V_{max}(J_i)\}$. Considering the fact that $Length(r_l)$ of the resource r_l is also constant, the optimal duration p_{ij} of the operation o_{ij} on the resource r_l can be determined as $p_{ij} = \frac{Length(r_l)}{V_{ij}}$. If a train stops at the resource, it will consume additional time t_{stop} for stopping (slowing down). That additional time is constant for all trains of the same category, i.e. $t_{stop} = t_{stop}(cat_i)$. Therefore, in case of planned stopping, additional time for stopping can be included into the optimal duration of operation p_{ij}. Similarly, the duration of the operation at the next device found on route can be increased by additional time t_{start} for starting (accelerating), i.e. the duration of the next operation $p_{i(j+1)}$ will be extended for $t_{start}(cat)$. For this reason thereafter, when formulating other problem constraints, when duration p_{ij} of the operation o_{ij} of the train J_i is extended by $t_{stop}(cat_i)$ or $t_{start}(cat_i)$, we will always consider that $t_{stop}(cat_i) = t_{start}(cat_i) = 0$ if the train J_i is planned to stop on resource $\phi_i(j)$. Otherwise, if train J_i stops in an unplanned manner, these additional times are larger than zero.

Since the model allows a train stopping on station tracks only, the duration of movement on them can be twofold: optimal, if there are no unplanned stops, or prolonged, for the stopping time. Similarly, the duration of movement on entry-exit resource when exiting the station can be optimal or prolonged for the time necessary for acceleration.

The rule on speeds is not explicitly formulated in constraint component implementation, but is used instead to calculate the operations' durations, which represent inputs for the job shop scheduling problem. However, the fact that limiting train speeds is one of the basic rules in railway traffic justifies our decision to define it in such a way.

Rule on stopping

The model assumes that all trains can only stop (in a planned or unplanned manner) in stations, i.e. in resources of type *st*. The rule which prohibits stopping in resources of type *bs* and *ee* can be formulated as

the following constraint: let $J_i \in J, \phi_i(j) = r_l, 1 \leq j \leq k_i - 1$, where $Type(r_l) = bs$ or $Type(r_l) = ee$ and let $\bar{d_{ij}}$ be the actual start time of the operation o_{ij}. Then, $\bar{d}_{i(j+1)} = \bar{d_{ij}} + p_{ij}$. Therefore, the completion time of the operation on the current resource is equal to the actual start time of the operation on the next resource. In other words, the sequence of operations which represents train movement between stations must be performed without waiting. Naturally, for operation o_{ij} performed on a resource of type st, the usual conjunctive constraint applies: $\bar{d}_{i(j+1)} \geq \bar{d}_{ij} + p_{ij}$.

Rule on occupying and making unary resources available

It is obvious that while in the system, the train must occupy at least one resource. Therefore, a resource is free not when its processing is finished, but when processing of the next job operation begins, plus time to release the resource. Generally, the time to release the resource $t_{release}$ depends on the train length (one property of the category) plus the overlap constant and whether a specific train stopped on a specific resource or not. Formally, let $J_p, J_q \in J$ be any two jobs such that $\phi_p(j_p) = \phi_q(j_q) = r_l$ and $Capacity(r_l) = 1$. Jobs $J_p, J_q \in J$ have operations o_{pj_p} and o_{qj_q} that should be performed on the same unary resource. Then,

$$((\bar{d}_{p(j_p+1)} + t_{release}(cat_p, standing(J_p, r_l)) \leq \bar{d}_{qj_q}) \vee (\bar{d}_{q(j_q+1)} + t_{release}(cat_q, standing(J_q, r_l)) \leq \bar{d}_{pj_p}))$$

where the value $standing(J_p, r_l)$ is 1 if the train J_p stopped on resource r_l, and 0 otherwise. Considering that stopping on resources for which $Capacity(r_l) = 1$ is forbidden by the preceding rule, $standing(J_p, r_l)$ this rule always assumes the value of 0. However, to be able to use the same function for formulating other rules of the model, we used this formulation.

Rule on occupying and making station tracks available

Station tracks are places in which trains stop (wait) for overtaking and crossing. The number of trains that can be in the station at the same time is limited by its capacity. The trains present in the station are all those whose processing is underway (movement or planned dwelling), but also those that finished processing on that device, but have not yet started processing on the next resource. Let $Type(r_l) = st$ and $Capacity(r_l) = k$. Let us consider all subsets of the jobs set $S \subseteq J$, such that $\mid S \mid = k + 1$ and for each $J_p \in S$ there is j_p such that $\phi_p(j_p) = r_l$. In another words, resource r_l is on the route of each train from S. It is clear that in every such train set S, there must exist at least two trains J_p and J_q for which the following applies:

$$\left(\bar{d}_{p(j_p+1)} + t_{release}(cat_p, standing(J_p, r_l)) \leq \bar{d}_{qj_q}\right) \vee \left(\bar{d}_{q(j_q+1)} + t_{release}(cat_q, standing(J_q, r_l)) \leq \bar{d}_{pj_p}\right)$$

where the value $standing(J_p, r_l)$ is 1 if the train J_p stopped on resource r_l, and 0 otherwise. Therefore, at least two trains from set S occupy the station of capacity k not at the same time.

Rule of sequencing

In our model trains follow the so-called block distance, i.e. between each two trains sequencing on open-line section between stations, there is at least one block section and behind it another resource in the direction in which trains move. Formally, let $J_p, J_q \in J$ are such that $\phi_p(j_p) = \phi_q(j_q) = r_l, \phi_p(j_p+1) = \phi_q(j_q + 1) = r_v$ for $1 \leq j_p \leq k_p - 2, 1 \leq j_q \leq k_q - 2$ and $Type(r_l) = bs$. Then,

$$\left(\bar{d}_{qj_q} \geq \bar{d}_{p(j_p+2)} + t_{free}(cat_p, standing(J_p, r_v))\right) \vee \left(\bar{d}_{pj_p} \geq \bar{d}_{q(j_q+2)} + t_{free}(cat_q, standing(J_q, r_v))\right)$$

Notice that from the condition $\phi_p(j_p) = \phi_q(j_q) = r_l$ and $\phi_p(j_p + 1) = \phi_q(j_q + 1) = r_v$ it follows that $direction(J_p) = direction(J_q)$. The consequence of this constraint is arrival of the trains from the same track to the same station at different times, where the minimum interval between their arrivals can be calculated knowing the lengths of resources and train speeds. Also, the consequence of this rule and the rule which forbids stopping on resources of type ee is that the necessary condition for a train to leave the station is that the first type bs resource is free, and following it the next resource should also be free. Implicitly, open-line section with one block section does not allow train sequencing.

Rule of crossing

The rule is related to $J_p, J_q \in J$ so that direction(J_p)≠direction(J_q). The safety regulations specify that at least t_c time units must elapse since the moment of making the line available until the moment of exit of the train in opposite direction on the same line. Let $\phi_p(j_p) = \phi_q(j_q) = r_l$, $\phi_p(j_p + 1) = \phi_q(j_q - 1) = r_v$ for $j = \{2, \ldots, k_p - 1\}$, $j \in \{2, \ldots, k_q - 1\}$ and $Type(r_l) = st$. It is clear that $Type(r_v) = ee$. Hence, $(\bar{d}_{p(j_p+1)} \leq \bar{d}_{q(j_q-1)}) \vee (\bar{d}_{qj_q} + t_c \leq \bar{d}_{p(j_p+1)})$.

Rule of planned dwelling in a station

For a given railway network, all trains belonging to the same category have planned stops on the same stations and have the same planned dwelling in all stations. The duration of planned train dwelling J_p is determined by its category, i.e. $t_{pw} = t_{pw}(cat_p)$. Let $\phi_p(j_p) = r_l$, for $1 \leq j_p \leq k_p$ and $Type(r_l) = st$. Then, $\bar{d}_{pj_p} + p_{pj_p} + t_{pw}(cat_p) \leq \bar{d}_{p(j_p+1)}$. In our model, the planned dwelling of train in station (for passenger operations or for traffic reasons) is included into the ideal duration of operation on a resource of type st.

Rule of non-simultaneous arrivals to station

This rule is applied on $J_p, J_q \in J$, such that $direction(J_p) \neq direction(J_q)$. Actual safety rule specifies that at least t_{ns} time units must elapse from a stopped train in a station to allow a train from the opposite direction to enter the station, i.e. to occupy entry-exit resource preceding station track in which the train is entering. Let $\phi_p(j_p) = \phi_q(j_q) = r_l$, for $1 \leq j_p \leq k_p, 1 \leq j_q \leq k_q$, and $Type(r_l) = st$. Then, we must distinguish cases when either j_p or j_q is equal to 1 and when both are different from 1. The situation when both are 1 is impossible because it would mean that two trains from different directions start moving in the same station and apply for the same open-line section. Hence,

$$((j_p = 1) \wedge (j_q \neq 1)) \Rightarrow \left((\bar{d}_{q(j_q-1)} \geq \bar{d}_{pj_p} + p_{pj_p} + t_{stop}(cat_p) + t_{ns}) \vee (\bar{d}_{qj_q} + p_{qj_q} + t_{ex}(r_l) \leq \bar{d}_{pj_p})\right), i.e.$$

$$((j_p \neq 1) \wedge (j_q \neq 1)) \Rightarrow \left((\bar{d}_{q(j_q-1)} \geq \bar{d}_{pj_p} + p_{pj_p} + t_{stop}(cat_p) + t_{nd}) \vee (\bar{d}_{p(j_p-1)} \geq \bar{d}_{qj_q} + p_{qj_q} + t_{stop}(cat_q) + t_{nd})\right)$$

Extra time $t_{ex}(r_l)$ depends on whether a particular station is physically the end of the line (trains depart from it or finish the route in it) or it is in place where one or more single-track and/or double-track lines join. When the station is the physical end, $t_{ex}(r_l) = 0$.

These constraints are identical both for initial scheduling and for rescheduling.

3. The third group of rules are constraints of the rescheduling model. These constraints differentiate the rescheduling problem from the initial scheduling problem. Our model formulates one of such constraints:

The rule of entering the system without waiting

The modeling assumption that each job J_i must be taken for processing at the moment of generation, can be simply expressed by: $\bar{d}_{i1} = d_i$. This actually means that the jobs cannot be "piled up" before entering the system. This assumption is extremely reasonable for the case of rescheduling, where the railway network under consideration is only a small fragment of real railway network. The moment of generation is actually the moment of train entering in the part of real railway network under consideration. The originating and destination stations in the model are in most cases only the intermediate stations in the real system.

4. Finally, a number of special constraints have been considered which can be of practical importance in operational control and which can be included selectively in the constraint component, in relation to requirements put to the rescheduling system. These constraints offer a possibility of planning very specific traffic situations. Our model deals with:

Rule of simultaneous stop-over in the station

Sometimes there is an interest for two trains to "meet" necessarily in one of the stations of the system. The rule of a simultaneous stop-over in the station should provide for trains $J_p, J_q \in J$, to stop simultaneously in a station at least for t_m time units (e.g. due to the planned overtaking, crossing or changing trains by passengers). The station is specified by the resource $Type(r_l) = st$, $\phi_p(j_p) = \phi_q(j_q) = r_l$, for $j_p \in \{1, 2, \ldots, k_p - 1\}$, $j_q \in \{1, 2, \ldots, k_q - 1\}$. Let $t_{stop}(cat_i)$ be additional time for stopping train J_i. Then, $\min(\bar{d}_{p(j_p+1)}, \bar{d}_{q(j_q+1)}) - \max((\bar{d}_{pj_p} + p_{pj_p} + t_{stop}(cat_p)), (\bar{d}_{qj_q} + p_{qj_q} + t_{stop}(cat_q))) \geq t_m$.

Rule of time distance between the completion of one and generation of another job

For two trains $J_p, J_q \in J$, it can be required that the train J_p completes operation on a resource $r_i, Type(r_l) = st, \phi_p(k_p) = \phi_q(1) = r_l$, at least t_{dis} time units before planned generation of the job J_q. Such constraint has sense because then the same locomotive used for train J_p can be used for train J_q, where t_{dis} time units is necessary for locomotive setup operations. Hence,

$$\left(\bar{d}_{pk_p} + p_{pk_p} + t_{stop}(cat_p)\right) - d_q \geq t_{dis}.$$

Rule of unavailability of resources

The purpose of this rule is to forbid the use of specified resource r_l (for example, for planned maintenance) in the time interval $[t_1, t_2]$. Hence, for every J_p such that there exists j_p for which $\phi_p(j_p) = r_l$, for $1 \leq j_p \leq k_p - 1$ the following must be true:

$$\left(\bar{d}_{p(j_p+1)} + t_{release}(cat_p, standing(J_p, r_l)) < t_1\right) \vee \left(\bar{d}_{pj_p} > t_2\right)$$

Rule of special separation

In some circumstances a constraint forbidding the use of resource r_l in t_{ss} time units after some "special train" J_p has released the resource can make sense. Therefore, let J_p, J_q be such that $\phi_p(j_p) = \phi_q(j_q) = r_l$, for $1 \leq j_p \leq k_p, 1 \leq j_q \leq k_q$. Then:

$$\left(\bar{d}_{p(j_p+1)} + t_{release}(cat_p, standing(J_p, r_l)) + t_{ss} \leq \bar{d}_{qj_q}\right) \vee \left(\bar{d}_{qj_q} < \bar{d}_{pj_p}\right)$$

Therefore, a "special train" is followed with a special distance. Similarly, a constraint can be defined formulating the distance by which a "special" train follows an "ordinary" train.

4.3 Problem objectives

The most common criterion with the job shop scheduling problems is minimizing the makespan. However, train scheduling is interested in the criteria taking into consideration the delays and different priorities of different train categories. Thus, we developed optimization models for the following seven objective functions:

1. minimization of the maximum delay $T_{max} = \max(T_1, T_2, \ldots, T_n)$. Although this criterion minimizes the maximum delay, several trips may suffer disturbances. The criterion is acceptable in situations when passenger trains prevail for scheduling.
2. minimization of the maximum weighted delay $WT_{max} = \max(w_1T_1, w_2T_2, \ldots, w_nT_n)$ may be an interesting criterion under the mixed traffic conditions. The weights are usually the same for all trains of the same category.
3. minimization of the total delay for trains $D = \sum_{i=1}^{n} T_i$
4. minimization of the total weighted delay for trains $WD = \sum_{i=1}^{n} w_iT_i$
5. minimization of the maximum slack of trains in stations, i.e. the minimization of the function $S_{max} = \max\{\bar{d}_{ij} - (\bar{d}_{ij} + p_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k, Type(\phi_i(j)) = st\}$. Considering that the absolute compliance with the original timetable corresponds to the situation that $S_{max} = 0$, this criterion supports the idea of maximum resemblance of the rescheduled timetable to the original.

6. minimization of maximum complete time of all jobs (makespan) $C = \max\{C_1, C_2, \ldots, C_n\}$, useful for situations in which trains should leave the fragment of railway network under consideration as soon as possible. In case of rescheduling, this objective function supports localization of disturbances.

7. minimization of the number of late jobs $|WJ|$, where $WJ = \{J_i \in J \mid C_i - c_i > 0\}$.

Which objective function will be chosen in some particular case can depend on the hour, weekday (working day, weekend), or the season in which the rescheduling is applied. The choice of the objective function can also depend on extraordinary circumstances in the near past, infrastructure operations planned for the near future, statistical analysis of the system history. Our solution leaves the dispatcher the freedom to choose the most appropriate objective function, i.e. the optimization model associated with it, based on the current traffic state and his own judgment. Some authors describe also other optimization criteria (e.g. minimizing the number of cancelled trains) which are not treated in this paper.

5 Solving train scheduling problem by constraint programming approach

Constraint Programming (CP) approach attempts to reduce the gap between the high-level problem description and an algorithm that solves the problem. CP attracts the experts' attention in various fields, since many real-world problems can be represented by Constraints, where the Satisfaction of these constraints gives a solution for the Problem in question [21].

The CP paradigm focuses on manipulation of variable domains and relations between these variables expressed through constraints. These variables are actually the *decision variables*, but they will be referred to hereinafter as *variables*, in short.

In real application there is an interest of determining the quality of the solution found. It is also sometimes an aim to find the best, optimal solution. The CS problem is therefore expanded by a function that joins to each solution a numerical measure of its efficiency – by an *objective function*.

If we consider the start times of activities $\bar{d}_{ij}$ in formulation of job shop problem as decision variables, to which the finite domains are added, and the conjunctive, disjunctive and special constraints as a set of constraints, it is clear that the job shop scheduling problem is a CSP. Supplemented by an objective function, it grows into a CS Optimization Problem. Hence, our train scheduling problem may be formulated as a CSP, i.e. CSOP.

The support in finding the solution in CP paradigm is offered by consistency methods and search strategies. The consistency methods make the constraint propagation through variable domains. In this way the variable domains are bound and the search space reduced. The complete search is applied if the aim is to find an optimal solution; if the aim is a "good enough" solution in the limited time, a local search combined with constraint propagation is applied.

The constraint programming approach has become an appealing technique only after the appearance of commercial CP tools. Within the available CP tools, the consistency methods and search strategies have been implemented as their inference mechanisms.

5.1 The choice of strategy, policy, and method of rescheduling

Following the ideas presented in [30], the essential steps in implementation of rescheduling are the choice of *factors*, *strategy*, *policy,* and *method of rescheduling.*

The rescheduling is activated after the recognition of the rescheduling factor. The rescheduling factor – the disturbance, in our case study is an unplanned forwarding of a train to the station that is equipped with the rescheduling system.

The strategy is necessarily predictive-reactive since there is an original timetable.

The choice of policy depends on the assessment of the minimum time spacing between the consecutive rescheduling factors and the expected run time for the rescheduling procedure. In general, the policy may be periodic, event-driven, and hybrid.

The event-driven policy is not suitable in case of frequent rescheduling factors, because the system would be in permanent rescheduling state. Junctions, i.e. places where tracks merge or diverge on an open line, allow for trains J_p and J_q to be directed from different stations s' and s'' to station s in very close (and possibly identical) moments t_p and t_q following one another to arrive from the same track into station s. Figure 3 demonstrates the sequence of trains of the first and the fifth category directed to Krnjaca station (from stations Beograd Dunav and Pancevacki most) with time distance of only 5 seconds. For this reason we chose the hybrid policy.

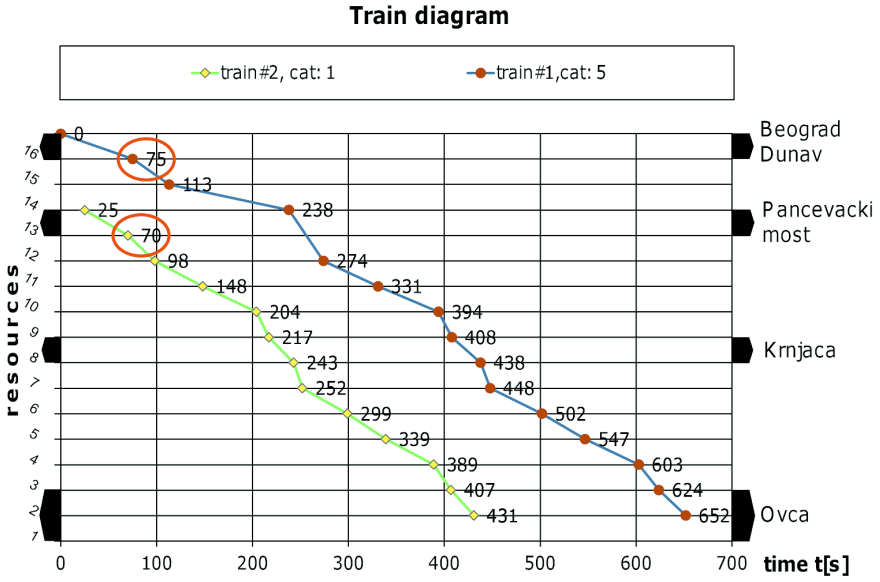

FIG.3. Possible frequent rescheduling factors – directing trains to Krnjaca station.

As far as the choice of the method is concerned, it is clear that due to time limit for rescheduling implementation, the focus should be on partial rescheduling methods.

The global rescheduling procedure is represented by the pseudo-code given in Figure 3. In effect, an infinite loop is in question: a disturbance is identified in the first part of the loop body, and processed in the second part.

The imperative clauses denote the procedures and function calls, where for some of them, due to the level of abstraction, the code may be omitted. All symbols in this algorithm are of a mnemonic character, and therefore, the comment is missing in a number of places.

The assumption is that the database `DB` comprises the initial schedule and network topology, as well as the updated dynamic data for the schedule implementation. Dynamic data in database `DB` will be tested periodically and if a disturbance is detected (i.e. deviation from the initial schedule bigger than the defined threshold), it triggers the rescheduling procedure. The available optimization models are incorporated in the modelbase `MB`. The procedure *SelectModel*, in accordance with a criterion which the user has chosen (from the set of seven offered), selects a scheduling model from the modelbase `MB`. The procedure *SelectPreparationModel* selects a preparation model corresponding to the chosen scheduling model. If the scheduling model which minimizes the maximum delay, total delay, maximum slack, makespan or the number of late jobs is chosen, then the preparation model is to minimize maximum delay. If the model which minimizes the maximum weighted delay or total weighted delay is chosen, then the preparation model is to minimize the maximum weighted delay.

Procedure *Trigger* verifies the contents of database `DB`, waiting for information on disturbance. If there is a piece of information on unplanned dispatching of one or more trains, the procedure returns a set of all jobs whose processing is underway at moment t- set `ActiveJobs`. Let `J` be the set of all jobs, including `ActiveJobs`, whose generation is planned up to the limit of the planning period t_g. It is clear that the planning period $[t, t_g]$ multiply exceeds the maximum flow time of jobs. Let $J_i \in$`ActiveJobs` be job dispatched to a station. The rescheduling of the remaining part of the job J_i will be carried out starting from that station. The expected train arrival to that station is a moment of actual generation $\bar{d}_i$ of the rest of the job J_i. This is clear because **Rule on stopping** forbids stops on open-line sections between stations, so if a disturbance finds some trains outside the stations, it can only affect them upon their arrival to the station towards which they are moving. For jobs $J_i \in J \backslash$`ActiveJobs`, d_i is the planned job generation subject to the initial schedule.

The procedure *Prepare* will be described in the part of the paper related to bound heuristics, and procedure *SeparateAndScheduleRelatedJobs* in the part describing separation heuristics. The procedure *SaveSchedule* stores the recovered schedule in the database, and procedure *PresentSchedule* visualizes the schedule in an adequate way.

input

DB, database

MB, base of scheduling models

forever

disturbance ←false

repeat

Trigger (DB, disturbance, ActiveJobs, J, t_g)

until disturbance **and** $t = k$·period, $k \in Z$

SelectModel (ActiveJobs, MB, DB, Model, MaxB)

SelectPreparationModel (MB, Model, PreparationModel)

Prepare (J, PreparationModel, Delays)

SeparateAndScheduleRelatedJobs (J, ActiveJobs, Model, Delays, Schedule)

SaveSchedule (DB, Schedule)

PresentSchedule (Schedule)

end forever

end procedure

FIG.4. Procedure *Reschedule*(DB, MB).

5.2 Heuristics

CP model consists of the following sections: declaration and definition of input data, definition of decision variables and their initial domains, definition of the objective function and the constraint component, specification of the user's search procedure and specification where and which results will be exported. The manufacturers of commercial CP tools claim that it is precise enough to formulate what the problem is (the constraint component and objective function, see Sections 4.2 and 4.3), and CP tools are capable of finding an optimal solution or a series of feasible solutions owing to inference incorporated algorithms, i.e. the specification of the user's search procedure is an optional part of the model. Experiments made with trial version of CPLEX Optimization Studio [15] prove unreliability of exclusive reliance on CP tools and their search algorithms. The process of arriving even up to the first solution is sometimes very time-consuming and as such cannot meet the rescheduling requirement. To improve the timing of the commercial tool, knowing the real problem in question, we formulated three classes of heuristics, called bound heuristics, separation heuristics, and search heuristics. These heuristics are a mechanism by which it is possible to apply custom-developed search strategies instead of the ones predefined by CP tool and thereby direct and confine search to the space where chances of quickly finding good solutions are high.

Bound heuristics

The aim of the bound heuristics is to limit the domains of decision variables and objective function in order to increase the search efficiency. Three types of bounds are proposed:

1. Initial bound: all variables take value from interval [origin, horizon], where the origin and horizon are assessed on the basis of the knowledge of the real problem:

   $$\texttt{horizon=origin} + \sum_{J_i \in J} \left(\sum_{o_{ij} \in J_i} p_{ij} + t_{stop}(cat_i) + t_{start}(cat_i) \right),$$

   where $\texttt{origin} = \min \{d_i \mid J_i \in J\}$, i.e.

   $$\texttt{horizon=origin} + \sum_{J_i \in J} (c_i - d_i) + t_{stop}(cat_i) + t_{start}(cat_i)),$$

   i.e. horizon is determined by the sum of duration of all operations, increased by additional times for stopping $t_{stop}(cat_i)$ and starting $t_{start}(cat_i)$ for every train $J_i \in J$.

2. The lower bound of objective function is estimated by a special procedure. The estimate is based on solving the preparation model that solves the conflict between two jobs in isolation, disregarding the consequences it might have on other jobs. The aim of procedure *Prepare* is to find the minimum delay to incorporate in a pair of jobs, if such pair of jobs is observed in isolation. Input arguments to *Prepare* procedure are the set of jobs to be scheduled and the already mentioned preparation model, while the output is `Delays` matrix. The element `Delays[i,k]` of matrix `Delays` is an optimal value of the objective function in solving the conflict between jobs J_i and J_k, using a preparation scheduling model. A set of inter-related jobs will be referred to as `RelatedJobs`. Procedure *Estimate*, called on every time before solving the scheduling model over the set `RelatedJobs` (see *SeparateAndScheduleRelatedJobs* procedure), on the basis of `Delays` matrix sets the lower bounds for the objective function. For objective functions which are not of a summary type, except for the number of late trains the following shall apply:

   $\texttt{LowerBound} = \max_{J_i, J_j \in RelatedJobs}\{\texttt{Delays}[i,j] \mid j > i\}$,

   and for summary type objective functions:

   $$\texttt{LowerBound} = \sum_{J_i, J_j \in RelatedJobs} \{\texttt{Delays}[i,j] \mid j > i\}.$$

   The lower bound of the number of late trains is a number of non-zero rows in `Delays` matrix. Clearly, the upper bound on the number of late jobs is the total number of jobs that are currently being scheduled.

   Since the `PreparationModel` (argument of *Prepare* procedure) solves the conflict between a pair of jobs in isolation, disregarding the consequences this might have on other jobs, it is very probable that such solution of a conflict situation includes a conflict between jobs that have not had it initially. It is also less probable that the solution of one conflict will necessarily resolve some other initial conflicts. Therefore, such heuristic for the estimation of the lower bound of objective function in most cases will help in avoiding the unfruitful ways of search, and only in a negligibly small number of cases we will give up very good solutions.

3. The upper bound of the objective function is dynamically bound during the search. For example, if the objective is to minimize the total delay, then, after finding a feasible solution with total delay D, we add the constraint: $\sum_{J_i \in J} \left(\vec{d}_{ik_i} + p_{ik_i} - c_i\right) \leq D$ to the model. The initial upper bound of objective function is the user-defined constant `MaxB`. Constant `MaxB` thus denotes the user requirement for the minimum schedule efficiency that is to be achieved.

Separation heuristics

Although dynamic job shop implies infinite number of jobs, it nevertheless processes only a finite subset of jobs at any given time interval. This is especially true for our train rescheduling problem. Safety constraints and limited number of traffic means available to railway operator give us the right to consider it as a phenomenon of "rarely generated jobs", which as such cannot quite mutually influence each other. Once the rescheduling factor is identified (i.e. when deviation from the initial schedule is bigger than the defined threshold), it is necessary to estimate the spatial and time point to which such deviation will propagate. Although a disturbance in the system usually has a chain reaction behavior, it is intuitively clear that it must be amortized in a reasonable time and space in order to reestablish the traffic according to the regular timetable.

Spatial bound of the system actually translates to shortening of traffic routes, while time bound reduces the number of jobs to be rescheduled. The goal of both types of bounds is to reduce the number of rescheduling operations so that the system is able to respond more quickly to an identified rescheduling factor. Hence, the aim is to separate and subsequently simultaneously schedule only those activities that can influence each other. Obviously, it is crucial to recognize those activities and it is the basic purpose of the procedure *SeparateAndScheduleRelatedJobs*.

If we want for a rescheduling system to support decision-making in one station only, it is clear that we must follow the events at least in all stations that are adjacent to the selected one (i.e. in the stations from which the trains can arrive or be sent to).

Let `RelatedJobs` be the set of jobs that must be rescheduled atomically (as a whole). Procedure *SeparateAndScheduleRelatedJobs* initially assigns the set `ActiveJobs` (the jobs whose processing is underway at the moment of disturbance) to the set of dependent jobs `RelatedJobs` that is recognized by *Trigger* procedure. After that the cycle starts with initially setting the values for `origin`, `horizon`, `LowerBound`, and `UpperBound` in a way described in **Bound heuristics**. Afterwards, the *SolveModel* procedure solves the chosen rescheduling

model. The obtained solution to the model can affect some jobs which have not yet started, and have not been scheduled at the moment, but are planned to be generated. `AdditionalRelatedJobs` set contains jobs which due to cascading effects among operations must be added to the set of dependent jobs; the task of one part of the procedure *SeparateAndScheduleRelatedJobs* is to find these additional jobs. It is clear that simple enlarging the `RelatedJobs` set does not make any sense; every subsequent iteration would include more and more jobs, which means that the number of operations and the time necessary to solve the model would increase enormously. For this reason, the task of *ExcludeActivities* procedure is to recognize such jobs in the `RelatedJobs` set, i.e. to recognize activities that may be considered definitely scheduled and to exclude them from the RelatedJobs set to prevent further scheduling. Also, if a job (with all its activities) is not completely scheduled, jobs from the `AdditionalRelatedJobs` set can possibly affect it only starting from some activity (i.e. station). Therefore, the job operations preceding this operation (station) can also be considered definitely scheduled and can be excluded from further scheduling. The `RelatedJobs` set is enlarged by the `AdditionalRelatedJobs` set. Until the `RelatedJobs` set is not empty, new values for `origin`, `horizon`, `LowerBound` and `UpperBound` are calculated and the loop body is repeated. The execution of the algorithm stops when the `RelatedJobs` set becomes empty. Generally, that can happen in two circumstances: when all jobs have been scheduled up to previously defined bound of the planning period t_g, or when the maximum complete time of jobs from the `RelatedJobs` set is less than the earliest moment of a generated planned job whose scheduling has not yet been considered (which actually means that all jobs from the `RelatedJobs` set will finish before the expected appearance of the first new job, which excludes the possibility of mutual influence). *SeparateAndScheduleRelatedJobs* procedure is preterm abandoned if *SolveModel* procedure does not find any feasible solution.

Search heuristics

It has already been mentioned that the variables in CP model are actual start times of activities $\bar{d}_{ij}$ and possibly some other additional variables. Each assignment of start times to activities which satisfies all existing constraints corresponds to finding a feasible solution. *SolveModel* procedure (called by *Prepare* and *SeparateAndScheduleRelatedJobs* procedures) calls *SetStartTimesOfActivities* procedure to find a feasible solution to the rescheduling problem by setting start times for activities.

Input arguments to *SetStartTimesOfActivities* procedure are jobs, i.e. their activities for which start times are set. Namely, in the case of activities with fixed duration, as in our case, the activity is scheduled if it has the start time assigned. The start time of each activity in CP approach is a decision variable with corresponding domain of interval type. We will look at the order of these variables that corresponds to the increasing lower bound of their domains d_{ij}. Such order supports the idea that activities with earlier planned start time d_{ij} are scheduled earlier. Also, the order of values within a domain (interval) is increasing. It is reasonable, because we want the actual start time of activity $\bar{d}_{ij}$ to be as close as possible to the planned start time d_{ij}. Initially, let d_0 be the earliest planned start time of all activities. All activities that can start at moment d_0, i.e. whose planned start time $d_{ij} = d_0$, are put into `SimultaneousActivities` set. It is possible that some subset of these activities may actually be scheduled to begin at moment d_0. To this purpose all non-empty subsets of the `SimultaneousActivities` set are formed. Logical procedure *CanStart* checks whether all activities of the given subset, according to the specified constraints, can actually start at d_0. Since our scheduling aims to schedule as many activities as possible as earlier as possible, the check starts from subsets with larger cardinality (larger number of elements). Note that in cases when constraints are not mutually exclusive, the very nature of the scheduling problem dictates that at least one activity will be scheduled to start at d_0.

Activities that are not scheduled to begin at the earliest possible time are put into `PostponedActivities` set and their earliest start time must be updated, i.e. the corresponding domains (intervals) reduced. For activities that do not belong to the `SimultaneousActivities` set, the earliest start time d_a is calculated. Also, for activities belonging to `PostponedActivities` set, the latest start time d_p is calculated. If $d_p < d_a$ for at least one of the postponed activities, the updated start time cannot be determined. This means that the current partial solution cannot be extended to the feasible solution. In such a case, backtracking must be performed by choosing a new subset of the `SimultaneousActivities` set and by trying to reschedule the activities of that set. If there are no subsets of the `SimultaneousActivities` set that have not been yet chosen, the scheduling problem does not have a solution (there is no feasible schedule).

If it is not true that $d_p < d_a$, the `ScheduledActivities` set is enlarged with the current subset; at the same time, this subset is excluded from the `PostponedActivities` set.

This process is repeated until the `ScheduledActivities` set reaches the `Activities` set (all activities are scheduled, i.e. a feasible solution has been found), or if there is no alternative after backtracking (no solution).

The procedure *SolveModel* calls the procedure *SetStartTimesOfActivities*, and thereafter, on the basis of a found feasible solution a new upper bound for objective function is set, and propagation of this constraint reduces the domains of decision variables. This is the preparation for the next call of the procedure *SetStart-*

TimesOfActivities. The procedure *SolveModel* stops further search if the upper bound of the objective function matches up with the lower bound, if rescheduling of start times of activities does not lead to a better value of objective function or if the procedure *SetStartTimesOfActivities* has not found a solution.

Such iterative process forms an optimal partial schedule, and if the described heuristics were successful, such partial schedules would form a "good enough" schedule within the limited time.

6 Experimental evaluation

For the purpose of experimental verification of the heuristic algorithms, a software system named *TimeRec* (Timetable Recovery) for train rescheduling has been designed and implemented. Windows application TimeRec, designed in Visual Basic, communicates with SQL Server database which contains the initial schedule and network topology, as well as the updated dynamic data concerning the schedule implementation. The chosen optimization model, created by using OPL modeling language, is executed in IBM ILOG CPLEX Optimization Studio [15]. The generated tabular schedule is saved in an Excel worksheet and in the database; the new schedule in the database acts as a trigger to start a procedure which visualizes the recovered schedule (Figure 5).

Experiments have been carried out on a fragment of real railway network (a part of the Belgrade Railway Node presented in Figure 1), with actual train categories operating there. The Belgrade Railway Node (BRN) is a very complex system dominated by a double-track line in the passenger traffic part, but also contains single-track lines, as well as junction points which have an additional negative impact on the organization and dispatching of the train traffic. The node is too complex and heterogeneous to be described by a single model. In this phase of the research we decided to study the part of the node presented in Figure 1 for the following reasons:

- The line Beograd centar – Pancevo, which uses this part of BRN, is a line with the highest passenger frequency in BRN, but also in a complete Serbian railway network. This is the route with the largest number of passenger trains.
- Pancevo is the major center of chemical, food, and manufacturing industry in Serbia. Pancevo port on the Danube (Pan-European corridor VII) is the origin and destination of a large number of freight trains. This single track line is the only connection of the city of Pancevo and its port to Belgrade, i.e. to Corridor X.
- Train scheduling problem on a single-track line is an especially complex problem because other than sequencing and overtaking trains moving in the same direction, the problem of crossing trains moving in opposite directions must also be solved.
- The network fragment in question is characterized by high degree of heterogeneity, with several categories of passenger and freight trains operating on it. The train structure (i.e. category structure) is highly variable during the day.
- This part of BRN contains the junction point "Pancevacki most" (resource no. 12 in Figure 1) from which three lines diverge/connect to: two single-track lines and one double-track line. This junction point can be classified as complex two-track junction with diamond crossing. On this section of BRN different technical systems of train organization are applied.
- Although this part of BRN is not large (maximum distance between stations is around 10 km), it is a bottleneck area in BRN and a frequent source of disturbances to regular timetable.
- There are different papers describing testing of models on real networks: [7] - central and southern Switzerland, [13] - dispatching area between Utrecht and Den Bosch, [16] - London Bridge area in the United Kingdom, [20] - Milan metro, which was our motivation to test our approach on a real-life example of the Serbian bottleneck area.

During model and software validation, the purpose of the first group of experiments was to determine the correctness and efficiency of the approach. We tested our software on a large number of timetables with conflicts. A number of timetables used were with traffic frequency significantly exceeding the real frequency of the network under consideration (BRN). The jobs (trains) are "piled up" on purpose to test the strength of the method. From the analysis of experiment results the following conclusions may be drawn:

- CPU time of schedule recovery depends on the number of activities and the number of conflicts;

FIG.5. Home page of TimeRec application and recovered schedule for the chosen criterion (minimization of the total delay D).

- solving of initial conflicts may bring up additional conflicts;
- in most cases (approximately 90%), the time performance is satisfactory for the purpose of operational reconstruction of the timetable. If the software does not find any feasible schedule within the given time interval (in our experiments for 30 seconds), the appropriate message will be generated;
- all generated timetables were feasible;
- heuristic nature of the approach has been demonstrated (in an insignificant number of cases the best known solution for the given objective function has not been found).

Figure 6 illustrates the recovered train schedule for infeasible schedule shown in Figure 2 for six objective functions (recovered schedule for the seventh objective function has already been shown in Figure 5). It is interesting to notice that all the found schedules are different from each other, i.e. depend on the selected objective function. Depending on the optimization criteria, we can notice a change of meeting the order of trains (trains #4 and #5, Figures d and e), change of stop/pass stations (train #6, Figures e and f) and the change of departing order (trains #2 and #3 in station Krnjaca, Figures b and c). All the discussed classes of constraints are taken into account. In this example, the initial timetable has 7 conflicts (6 disjunctive and 1 regarding the Rule of non-simultaneous arrivals to station), and the number of resolved conflicts is 9, regardless of the selected objective function. This is the expected behavior, because the cascading effect increases the number of conflicts.

Table 2 presents the value of the objective function and the corresponding CPU time necessary to find it, as well as values of candidate for objective functions reached for the chosen optimization model. All experiments were conducted on a PC with Intel Core 2 Duo P8400 2.27 GHz CPU and 3 GB RAM. Each model was run at least 10 times to determine the mean value of the CPU time. Any chosen objective function never takes values higher than the value it obtains for some other objective function. Therefore, in this example the heuristics served its purpose of directing the search into an area where “good“ solutions are found.

We can conclude that the software is absolutely correct and reasonably efficient because it is able to generate the recovered schedule for each of the selected models within a given time limit.

Further experiments consisted of comparing the recovered schedules generated by seasoned dispatchers to schedules generated by the rescheduling software TimeRec for this railway network. To this aim, we considered

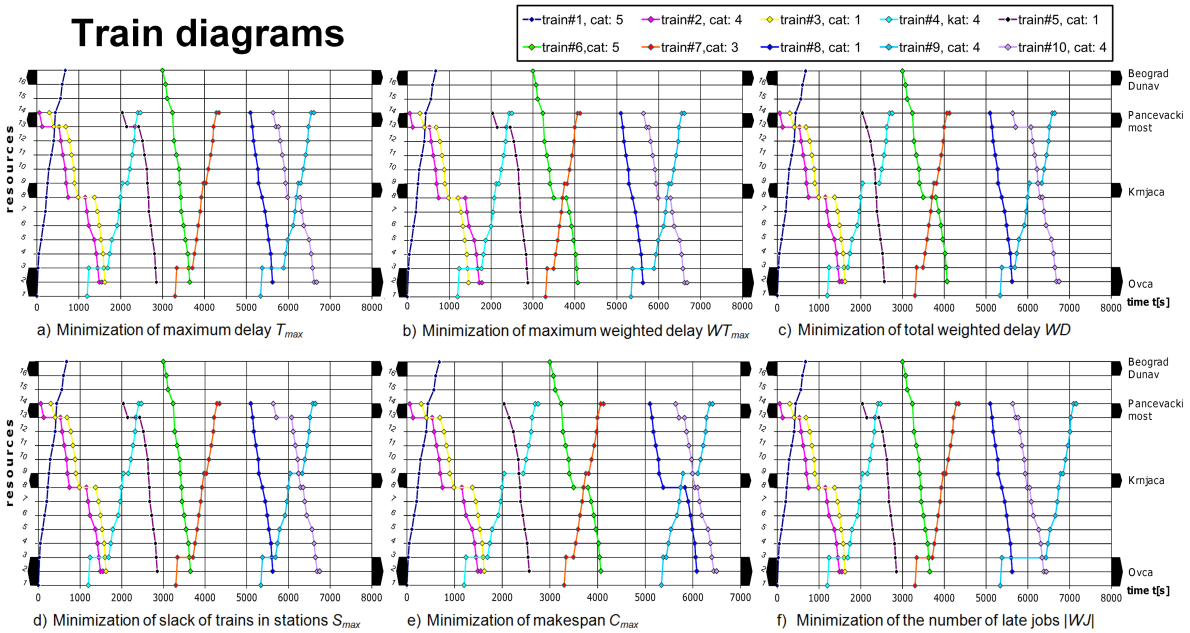

FIG.6. Minimization by different criteria.

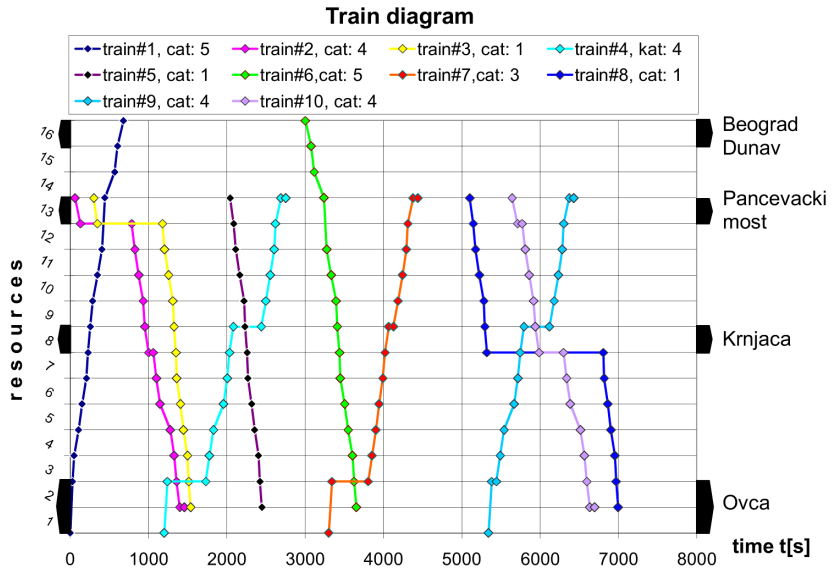

FIG.7. Train diagram generated by a dispatcher for the infeasible schedule shown in Figure 2.

actual realized train diagrams recorded during 35 working days in May/June 2010. The experiment considered 147 "traffic situations", i.e. implementations of movements of a set of n trains in a certain time period which do not influence some other train set. This set is actually before the mentioned set of inter-related jobs named `RelatedJobs`. At least one of n trains, actor in a "traffic situation", enters the studied system (i.e. is generated) with a certain delay in respect to the valid timetable. This can potentially cause a conflict situation so that an intervention by a dispatcher is necessary; such an intervention causes change in times of departure and arrival with respect to the ones planned by the timetable. Therefore, the actual realized traffic diagram in every "traffic situation" is a result of interventions and decisions made by a dispatcher. Dispatchers decide in a usual way, according to a dispatching rule FCFS (First Come First Serve): the resource is assigned to a train which first asks for it. When more trains require the same resource at the same time, the resource is assigned to the train with higher priority. TimeRec software was given the assignment to solve identical situations, where the chosen optimization model was the minimization of the total delay D. For example, for the traffic situation illustrated in Figure 2, dispatcher's solution is presented in Figure 7, and software solution in Figure 5. In this particular example, the total delay of trains in dispatcher's solution is 4256 s, and in software solution 3655 s (software solution is better by 14.12 %).

By analyzing the dispatchers' records for all 147 traffic situations and letting the software generate the solution for every case, the following conclusions have been drawn:

- on the average, the model reduces the total delay by 24.93%;
- the model suggests better solutions than those implemented by a dispatcher in 85.03% of cases, while in approximately 11% of cases both software and dispatcher found the same solution;
- in three cases, i.e. approximately 2%, the dispatchers' solutions from the records were slightly better than the software solutions (for less than 60 s). One possible cause for this may be that times of train arrivals to control points as recorded in the dispatchers' books and in the realized train diagrams are rounded to half a minute, so that the accumulation of rounding errors can cause such a result.
- in three cases, i.e. approximately 2%, software was not able to generate a feasible schedule. Further analysis demonstrated that this happened in cases of disruptions solved by cancelling trains, for which the software is not competent.

The experiments demonstrate that in a vast majority of cases the software is able to propose better schedules than the experienced dispatchers.

The rescheduling software system should enable the dispatcher to produce a faster and a better schedule. Therefore, the third group of experiments was aimed at validating software by dispatchers who are its potential users. The software was validated by 18 dispatchers in a considered network, who assigned grades in range 1 to 5 (the higher the better) by the following criteria:

1. user-friendliness;
2. the speed of generating the recovered schedule;
3. correctness (feasibility) of the proposed schedule;
4. clarity of graphical representation of the proposed schedule;
5. possibility of generating multiple alternative schedules;
6. reliability, i.e. the possibility to completely rely on schedules generated by the software.

The criterion no. 6 got the lowest average grade - 3.5; obviously, human dispatchers rely on their own experience and training. The highest average grade was given to criterion no. 5 - 4.6; dispatchers highly appreciate the possibility to have as much as seven potentially different solutions to the same disturbance.

This was a very important discovery for the authors of this paper. Namely, the aim of the research in the cited literature is to optimize by one or two criteria, rarely by three criteria. The most common criterion is the minimization of the delay (total, maximum, or weighted) of trains, passengers or freight, minimization of the deviations from the original timetable, minimization of secondary delays. Our experiments proved that for the highly congested network characterized by high degree of traffic heterogeneity and variability of train structure the possibility of choosing/changing the objective function and comparing the generated recovered schedules is highly desirable. For example, the left-hand part of Figure 8 represents the rescheduling plan when the criterion is minimization of the maximum delay, where values reached are $|WJ|$=5, (i.e. all trains are late), $Tmax$=432 s for train #9. The right-hand part of Figure 8 illustrates the rescheduling plan if the criterion is minimization of

TABLE 2. Results of optimization models applied to infeasible schedule shown in Figure 2.

Reached values of candidates for objective functions								
Objective function	T_{max}	WT_{max}	D	WD	S_{max}	$makespan$	$\|WJ\|$	CPU time [s]
T_{max}	918	3672	3586	10064	450	6681	8	21.99
WT_{max}	920	3024	3901	9999	578	6681	9	21.38
D	918	3672	3655	10538	456	6499	8	24.34
WD	918	3672	3882	9661	392	6751	9	27.08
S_{max}	918	3672	3690	10272	392	6751	8	22.43
$makespan$	918	3672	3853	10515	456	6499	9	26.99
$\|WJ\|$	992	3672	3889	10670	992	7156	7	20.90

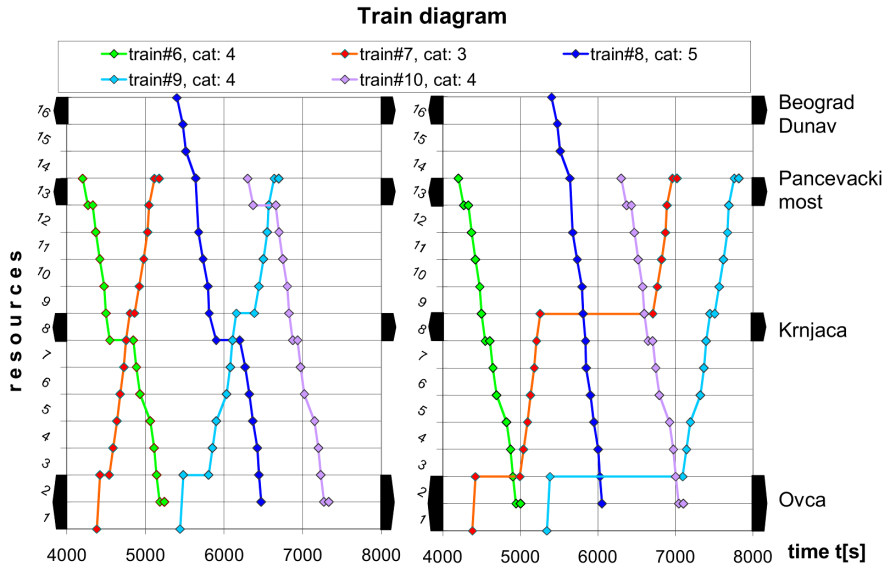

FIG.8. Two rescheduling plans for the same conflicting timetable.

the number of late trains where $|WJ|$=2, $Tmax$=1904 s for train #7. These solutions are very different, which dispatchers as potential users found very interesting.

TimeRec is a complete decision-support system that allows the user to:

- choose an optimization criterion in a user-friendly way (by choosing a drop-down menu item, Figure 5);
- find a recovered schedule in a very short time (less than 30 seconds) in more than 90% of cases;
- represent the generated schedule in the form of a diagram common to railway practice (train diagram, Figure 5).

Even if an experienced dispatcher does not want to rely completely on solutions proposed by the software, there is always enough time to consider a few proposed schedules and make the best decision in a given traffic situation (changeable structure of trains, peak hours, weekend hours, congested network, special demands). To our knowledge, this is one of the main contributions of the proposed solution that distinguishes it from solutions described in available literature.

7 Conclusion

This paper presents a very realistic railway transport model. Namely, the actual line-side signals that limit the resources have been taken into account, different train categories have got different priorities, junctions may exist on open-line sections, etc. The train rescheduling problem has been formulated and solved as the

Constraint Satisfaction Optimization Problem. We considered adequate the optimization criteria that take into consideration the tardiness and established the priorities between different train categories (e.g. maximum delay, total delay, maximum weighted delay). The special value of our solution is the possibility offered to a dispatcher to choose the most suitable optimization criterion from the set of seven available ones. This is especially important in circumstances when the train structure is highly variable in time. In order to improve the time performance of available constraint programming tool and to meet the rescheduling requirements, three classes of heuristics seeking the solution simultaneously have been proposed. They are referred to as separation heuristics, bound heuristics, and search heuristics. For the purpose of experimental verification of the proposed approach, a software system for train rescheduling has been designed and implemented. The experiments carried out on a fragment of real railway network (a part of the Belgrade Railway Node), with real train categories operating in that area, and with real possible disturbances, have proven the validity of our approach to solving problems of operational railway traffic control. If strict time limits are relaxed and the size of the problem increased, the method is capable of solving the problems of initial train scheduling on a real network up to the optimization within reasonable time. Also, by solving a difficult problem of train scheduling, we have paved the way for solving a whole series of problems, the core of which is the train scheduling, e.g. timetable preparation, determination of economically acceptable capacity utilization interval, the estimate of stopping and waiting of trains for traffic reasons, projection of investment activities results, identification of bottlenecks in infrastructure, the choice of a possible solution of a conflict point, research on allocation of block sections and line-side signals, etc. For these reasons, our future work will be directed to upgrading the existing software in several directions.

Acknowledgements

This work is partially supported by the Ministry of Education and Science of the Republic of Serbia, Project No 36012.

References

[1] R. Acuna-Agost, P. Michelon, D. Feillet, and S. Gueye, A MIP-based local search method for the railway rescheduling problem, *Networks* 57(1) (2011), 69 - 86.

[2] J. Blazewicz, W. Domschke, and E. Pesch, The job shop scheduling problem: conventional and new solution techniques, *Eur J Oper Res* 93 (1996), 1 - 33.

[3] V. Cacchiani, A. Caprara, and P. Toth, Non-cyclic train timetabling and comparability graphs, *Oper Res Lett* 38(3) (2010), 179 - 184.

[4] V. Cacchiani and P. Toth, Nominal and robust train timetabling problems, *Eur J Oper Res* 219(3) (2012), 727 - 733.

[5] V. Cacchiani, D. Huisman, M. Kidd, L.G. Kroon, P. Toth, L.P. Veelenturf, and J.C. Wagenaar, An Overview of Recovery Models and Algorithms for Real-time Railway Rescheduling, Transport Res B-Meth 63 (2014), 15 - 37.

[6] X. Cai, and C. J. Goh, A fast heuristic for the train scheduling problem, *Comput Oper Res* 21(5) (1994), 499 - 510.

[7] G. Caimi, M. Fuchsberger, M. Laumanns, and K. Schüpbach, Periodic railway timetabling with event flexibility, *Networks* 57(1) (2011), 3 - 18.

[8] A. Caprara, L. G. Kroon and P. Toth, "Optimization Problems in Passenger Railway Systems", *Wiley Encyclopedia of Operations Research and Management Science*, Vol 6, Wiley, 2011, pp. 3,896 - 3,905.

[9] K. Chiu, C. M. Chou, J. H. M. Lee, H. F. Leung, and Y. W. Leung, A constraint-based interactive train rescheduling tool, *Constraints* 7(2) (2002), 167 - 198.

[10] J-F. Cordeau, P. Toth, and D. Vigo, A survey of optimization models for train routing and scheduling, *Transport Sci* 32(4) (1998), 380 - 404.

[11] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, Bi-objective conflict detection and resolution in railway traffic management, *Transport Res C-Emer* 20(1) (2012), 79 - 94.

[12] D'Ariano, F. Corman, D. Pacciarelli, and M. Pranzo, Reordering and local rerouting strategies to manage train traffic in real-time, *Transport Sci* 42(4) (2008), 405 - 419.

[13] D'Ariano and M. Pranzo, An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances, *Netw Spat Econ* 9 (1) (2009), 63 - 84.

[14] S. Feng, L. Xin-Hua, J. Qin, and L.B. Deng, Earliest conflict optimal method for train operation adjustment on single track railway, China Railway Science, Chinese Academy of Railway Sciences, 100081, China, (2005), 106 - 113.

[15] IBM CPLEX (2012). CPLEX Optimization Studio. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio. Accessed online 20. March 2012.

[16] Khosravi, J. A. Bennell and C. N. Potts, Train Scheduling and Rescheduling in the UK with a Modified Shifting Bottleneck Procedure, *Proc 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'12)*, D. Delling and L. Liberti (Editors), pp. 120–131, OpenAccess Series in Informatics, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2012. http://drops.dagstuhl.de/opus/volltexte/2012/3708/pdf/13.pdf

[17] P. Kreuger, M. Carlsson, J. Olsson, T. Sjoland, and E. Astrom, Trip scheduling on single track networks – the TUFF train scheduler, Workshop on Industrial Constraint Directed Scheduling, 1997, pp. 1 - 12.

[18] C. Liebchen, The First Optimized Railway Timetable in Practice, *Transport Sci* 42(4) (2008), 420 - 435.

[19] R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan, Railway track allocation: models and methods, *OR Spectrum* 33(4) (2011), 843 - 883.

[20] C. Mannino and A. Mascis, Optimal real-time traffic control in metro stations, *Oper Res* 57(4) (2009), pp. 1,026 - 1,039.

[21] K. Marriott, and P.J. Stuckey, *Programming with Constraints: An Introduction*, The MIT Press, Cambridge, 1998.

[22] M. Mazzarello, E. Ottaviani, A traffic management system for real-time traffic optimization in railways, *Transport Res Part B-Meth* 41(2) (2007), 246 - 274.

[23] S. Mladenovic and M. Cangalovic, Heuristic approach to train rescheduling, *Yugoslav journal of operations research* 17 (2007), 9 - 29.

[24] E. Oliveira and B.M. Smith, A hybrid constraint-based method for single-track railway scheduling problem, Rep 2001.04. School of Computing, University of Leeds, 2001.

[25] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Springer, New York, 2008.

[26] J. Rodriguez, A constraint programming model for real-time train scheduling at junctions, *Transport Res Part B-Meth* 41(2) (2007), 231 - 245.

[27] N. Tomii, Y. Tashiro, N. Tanabe, C. Hirai, and K. Muraki, "Train rescheduling algorithm which minimizes passengers' dissatisfaction", *Innovations in Applied Artificial Intelligence*, *Lect Notes Comput Sc* 3533, Springer Verlag, 2005, pp. 829 - 838.

[28] J. Törnquist, "Computer-based decision support for railway traffic scheduling and dispatching: a review of models and algorithms", *Proc 5th Workshop on Algorithmic Methods and Models for Optimization of railways (ATMOS'05)*, L.G. Kroon and R. H. Möhring (Editors), 2005, http://drops.dagstuhl.de/opus/volltexte/2006/659.

[29] J. Törnquist and J.A. Persson, N-tracked railway traffic re-scheduling during disturbances, *Transp Res B-Meth* 41(3) (2007), 342 - 362.

[30] E.G. Vieira, J.W. Herrmann, and E. Lin, Rescheduling manufacturing systems: a framework of strategies, policies and methods, *J Scheduling* 6 (2003), 39 - 62.