Yugoslav Journal of Operations Research
17 (2007), Number 1, 9-29
DOI: 10.2298/YUJOR0701009M

HEURISTIC APPROACH TO TRAIN RESCHEDULING

Snežana MLADENOVIĆ
Faculty of Transport and Traffic Engineering, University of Belgrade, Serbia
snezanam@sf.bg.ac.yu

Mirjana ČANGALOVIĆ
Faculty of Organizational Science, University of Belgrade, Serbia
canga@fon.bg.ac.yu

Received: December 2005 / Accepted: February 2007

Abstract: Starting from the defined network topology and the timetable assigned beforehand, the paper considers a train rescheduling in respond to disturbances that have occurred. Assuming that the train trips are jobs, which require the elements of infrastructure – resources, it was done by the mapping of the initial problem into a special case of job shop scheduling problem. In order to solve the given problem, a constraint programming approach has been used. A support to fast finding "enough good" schedules is offered by original separation, bound and search heuristic algorithms. In addition, to improve the time performance, instead of the actual objective function with a large domain, a surrogate objective function is used with a smaller domain, if there is such.

Keywords: Train rescheduling, job shop scheduling, constraint programming, heuristics.

1. INTRODUCTION

The train scheduling problem belongs to a category of NP-hard problems of combinatorial optimization [1, 2], and hence is complex for both modeling and solving. The train scheduling problem considered for a larger fragment of railway network, a longer planning period, and hence the higher number of trains is a part of designing the timetable carried out at the level of tactical planning. The assignment of train rescheduling is that on a smaller fragment of railway network, over a shorter planning period an operational reconstruction of timetable is made, in respond to disturbances that have arisen. The rescheduling may be considered to be a more difficult problem than an initial scheduling because additional requirements are imposed to it [4]: to find a solution in a given real time; to have a recovered schedule which will deviate from the initial one as little as possible; the solution if not optimal, to be at least "enough good" with respect to the assigned objective function, but also to other performances, etc.

However, only a few published papers deal with train rescheduling in real time. In fact, the current rescheduling systems test mostly if the solution proposed by the user is a feasible one, and not doing full schedule regeneration [3]. It can also be noted that authors simplify the scheduling problem in two ways: by simplification of the network structure and omitting and/or approximating constraints that govern the train movement [7, 10]. The basic aim of the research is to formulate the most realistic model and develop original heuristic algorithms, which in conjunction with constraint programming mechanisms in a suitable way are able to find an "good enough" solution of the train rescheduling problem within the limited time.

The rest of the paper is arranged as follows: Section 2 defines in a concise and exact way the railway network topology. As a case study because of its complexity, a single-track line scheduling problem was chosen. Namely, the train scheduling on a double-track line can be assumed as a relaxed problem of scheduling on a single-track line, there being no train crossing. In section 3, the problem of train scheduling on a single-track network is modeled as a job shop scheduling problem. Assuming that the train trips are jobs, which require the elements of infrastructure – resources, it was done by the mapping of the initial problem into a special case of job shop scheduling problem. The fourth, key section, deals with solving the problem by constraint programming approach. After the train scheduling problem defined as a constraint satisfaction optimization problem, a set of constraints, the corresponding optimization criteria, heuristics for accelerating reaching good solutions and concept of the surrogate objective function are discussed. For the purpose of an experimental test of the described heuristic methods, the first prototype of software system for train rescheduling has been constructed. In the fifth section, using the train rescheduling software, the proposed method is evaluated on the selected real examples. The final considerations and possible directions of further research are presented in the last, sixth section.

2. 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), numbers of entry and exit points of the facility and possibility of connection, we can distinguish three disjunctive subsets: block sections – set P ($\mathcal{Type}(r)$=*bs*, $\mathcal{Capacity}(r)=1$, if $r \in P$), entry-exit facilities – set U ($\mathcal{Type}(r)$=*ee*, $\mathcal{Capacity}(r)=1$, if $r \in U$), and station tracks – set S ($\mathcal{Type}(r)$=*st*, $\mathcal{Capacity}(r)>1$, if $r \in S$). Such a classification is close to the real one, and it is suitable for an exact formulation of constraints concerning the occupying and making available each of the resource classes.

On railway lines with two-way traffic the train movement directions are traditionally designated as **odd** and **even**. In further presentation, we shall assume that the train moves in odd direction if it runs from the entry to the exit points of the facility, or in even direction from the exit to the entry point of the facility. The railway network is built by connecting the exit points of one facility to entry points of other facility. Facilities from the set P have exactly one entry and exit point; facilities from the set S have mutually equaled, and still higher than one, numbers of entry and exit points. The entry and exit points of the facility are provided with signals controlling its occupation. Let function $\mathcal{I}$ add to each facility the number of its entry points, and function $\mathcal{O}$ add to each facility the number of its exit points.

Let $L = \{bs, st, ee\}$ be the set of possible facility types. The ternary relation μ is defined on set L such that $(x, y, z) \in \mu$ means that the facility of type y may be connected through one its entry point with the facility of type x and through one its exit point with facility of type z . The relation μ shall be represented by the set of triples:

$$\mu=\{(bs,bs,bs),(bs,bs,ee),(ee,bs,bs),(bs,ee,bs),(bs,ee,st),(st,ee,bs),\\ (bs,ee,ee),(ee,ee,bs),(st,ee,ee),(ee,ee,st),(ee,ee,ee),(ee,bs,ee),(ee,st,ee)\}.$$

The railway network N can be defined as a directed acyclic graph $N = (R, A)$, where nodes are resources from R, and arc $(r_p, r_q) \in A$ between resources r_p and r_q means that the exit point of the r_p facility is connected to the entry point of the r_q facility. The network is properly built if for each triple of connected facilities r_p , r_q , r_k where $(r_p, r_q) \in A$ and $(r_q, r_k) \in A$, goes that $(\mathcal{T}ype(r_p),\mathcal{T}ype(r_q),\mathcal{T}ype(r_k)) \in \mu$ and if in each node r_q at most $\mathcal{I}(r_q)$ arcs flow in, and at most $\mathcal{O}(r_q)$ arcs flow out.

The stations are modeled as resources of type *st,* which are, in accordance with relation μ , in connection with the resources of type *ee* through entry and exit points. Open-line sections between stations consist of one or more block sections (resources of type *bs*) between which bifurcation points may be found (resources of type *ee*). This makes possible for 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.

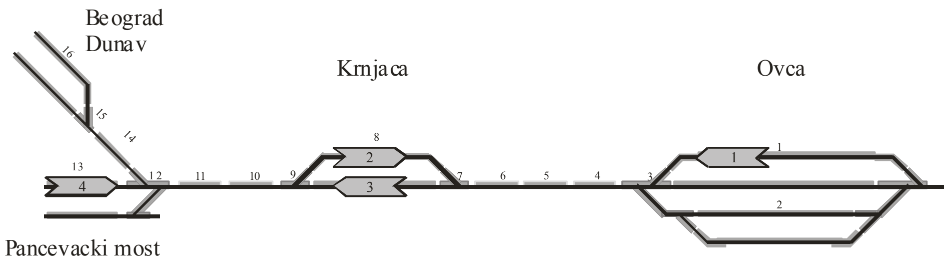

Figure 1: Real example of a railway network

3. SINGLE-TRACK TRAIN SCHEDULING AS A JOB SHOP SCHEDULING PROBLEM

The timetable is an entry into the operational railway control. The timetable specifies starting and final points of journey as well as the scheduled arrival and departure times for each intermediate station on 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 destination. This paper interprets the term route in somewhat modified way. The **route** is a sequence of facilities the train must cross on its journey from the origin to destination. A valid route in the network *N* is any path with nodes r_{l_1}, r_{l_2}, ..., r_{l_e} so that $\mathfrak{Type}(r_{l_1})=\mathfrak{Type}(r_{l_e})=st$, (for trains moving in odd direction), i.e. its inversion (for trains moving on even direction).

Instead of the arrival and departure times for each train and each facility on its route, we shall 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 of the train. Since we know the planned train arrival time to the first facility on the route – **planned train generation**, based on an ideal duration of occupation, we can assume that **the ideal train timetable** is known.

The train movement is a series of particular trips, operations of facilities occupation en route. Hence, each train is accompanied by a trip in a unique way. The trains 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 resources exceed their capacities, or when some of the imposed constraints have been disturbed regarding the train movement control. In a general case, solving of conflicts requires an introduction of delay into at least one of the conflict trips.

The scheduling model of interest in this paper is a dynamic job shop model [6, 11]. The model is a complex processing system with several resources and several operations, where each job has its inherent sequence of operations (activities) and inherent generation time.

The 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, $J = \{J_1, ..., J_n\}$ set of train trips considered as jobs and *N* is the railway network. Each train trip $J_i \in J$ is a series of k_i operations (activities) $J_i = (o_{i1}, ..., o_{ik_i})$. To each train trip J_i the "1-1" mapping $\phi_i : \{1, \quad 2, ..., k_i\} \rightarrow R$ is corresponded. ϕ_i 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. Sequence $\phi_i(1)$, $\phi_i(2)$, ..., $\phi_i(k_i)$ must represent one path on network *N* (when a *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 of $\phi_i(j)$ by the job J_i. Each train is joined its category cat_i, for which we shall assume to specify all fixed train attributes (type – passenger or freight, length, weight, speed, time for stopping, time for starting, etc.). The function w joins to each job J_i its priority w_i. Job priority depends on train category, the scheduling period, external events, etc.

The planned start time d_{ij} for each operation o_{ij} is equal to the earliest possible time, i.e. the earliest completion time of the preceding operation: $d_{ij} = d_{i(j-1)} + p_{i(j-1)}$, $2 \le j \le k_i$ 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 of J_i, then the tardiness T_i of job J_i 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, consist of finding the actual start time $\overline{d}_{ij}$, $\overline{d}_{ij} \ge d_{ij}$, for each operation o_{ij} of each of the jobs J_i avoiding conflicts, meeting additional constraints and optimizing the selected objective function. If the ideal timetable is a feasible one, then $\overline{d}_{ij} = d_{ij}$ for all operations of each job.

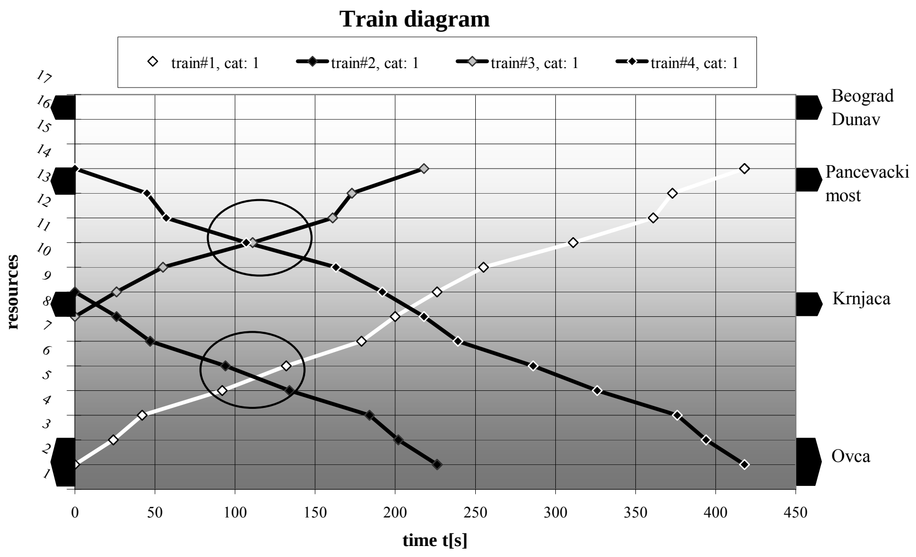

Figure 2: An infeasible ideal timetable with conflicts on open-line between stations Krnjaca – Ovca and Pancevacki most – Krnjaca

If trains in Figure 1 started the movement simultaneously and moved at the same speeds, the trains 1 and 2 would have a conflict on an open-line between stations Krnjaca – Ovca, and trains 3 and 4 on an open-line between stations Pancevacki most –

Krnjaca. Hence, the ideal timetable is not feasible in this particular case. The visualization of this conflict is given in Figure 2 in the form that is known in railway traffic as the **train diagram**. Actually, it represents the modified Gantt's diagram, where the modification consists of touching resources on *y*-axis, and bars are replaced by their diagonals, which symbolizes the train movement on the resource.

4. SOLVING TRAIN SCHEDULING PROBLEM BY CONSTRAINT PROGRAMMING APPROACH

4.1. Definition of the Train Scheduling Problem as CSO Problem

Constraint programming (CP) is a more recent approach in the field of programming languages, which attempts to reduce the gap between the problem description at a high level and algorithms implemented for its solving. One of possible CP definitions is: CP proposes the software architectures for simplifying the implementation of combinatory optimization algorithms. CP attracts the experts' attention in various fields, finding out that many problems in real world can be represented by constraints, where the satisfaction of these constraints gives a solution for the problem in question (CSP).

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

Formally, CSP is defined as a triple (V, D, C) [8], where: $V = \{v_1, ..., v_n\}$ is a finite set of variables presenting the problem, D is a function that joins to each variable in V its domain, i.e. $D(v_i) = D_i$ and C is a finite set of constrains. Constraint $c \in C$ between the variables v_{j_1}, v_{j_2}, ..., $v_{j_k} \in V$, $k \leq n$ is a subset of Descartes' product of their variable domains, i.e. $c(v_{j_1}, v_{j_2}, ..., v_{j_k}) \subseteq D_{j_1} \times D_{j_2} ... D_{j_k}$. The **CSP solution** is such an assignment of values from domains D_i to variables v_i, $i = 1, ..., n$ which satisfies all constraints from *C*. Let *S* be the set of all CSP solutions.

In real applications, 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 an objective function *f* that joins to each solution from *S* a numerical measure of its efficiency. Function *f* is defined as an arithmetical expression over variables in *V*. Formally, constraint satisfaction optimization problem (CSOP) is defined as a quadruple: *(V, D, C, f)*.

If any upper limit U_0 of function $f(S)$ is known (in the case its minimization), then the constraint $f(S) < U_0$ may be added to the set of constraints of CSP problem, i.e. the problem $(V, D, C \cup \{f(S) < U_0\}, f)$ is under consideration. A solution of this problem gives a new limit U_1, ($U_1 < U_o$ for the minimization problem), and now the problem $(V, D, C \cup \{f(S) < U_1\}, f)$ may be solved. Hence, CSOP can be solved incrementally until an optimal solution is reached.

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

A 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 bounded and the search space reduced. The complete search is applied if the aim is to find an optimal solution; if the aim is a solution "good enough" in the limited time, a local search is combined with constraint propagation.

The CP approach has become an appealing technology for planning and scheduling problems 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.

The motivation for choosing the CP approach in solving our problem is as follows:

- declarative nature of constraints in CP approach offers a comfort in formulating the numerous and complex constraints occurring in real train scheduling problem on a single-track railway network;
- the presence of commercial CP tools may significantly shorten the development time and length of the programming code of scheduling applications;
- separation of the constraint component from the search component offers a possibility to keep the constraints once formulated (these actually being regulations for train movements on the railway line which are relatively seldom changed), and to build the search component by considering the concrete objective function;
- possibility of dynamic modification of constraint set by adding new constraints in order to satisfy the current requirements;
- researchers' challenge to test a new approach to solving the train rescheduling problem! Namely, the train rescheduling on a single-track network has been a subject of research for the first author of this paper for a number of years. Thus, in [5, 9] the weighted priority dispatching rules and heuristic scheduling rules have been used, aiming at minimizing the weighted number of late trains. As opposed to dispatching rules permitting quick decision-making, but not taking into account the global information, the CP approach carries out a systematic search and as such, it is time-consuming, but also capable of finding better solutions, according to the assigned objective function. Before carrying out the research, it was not clear if time-consuming CP method may satisfy the time-limited train rescheduling.

4.2. Constraint Component

In order to solve the train rescheduling problem by CP approach, it is necessary to define the constraint component. The constraint component in our research consists of four classes of constraints.

1. As the train scheduling problem is formulated as 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 trains collisions and it exists in each railway system. In the carried out research a minimum set of eight **safety constraints** has been defined, covering all known regulations applicable in real train movement on a single-track line on the Serbia and Montenegro railway network. These constraints are designated as: Rule on speeds, Rule on stopping, Rule on occupying and making available unary resources, Rule of occupying and making available station tracks, Rule of sequencing, Crossing Rule, Rule of stop-over in a station and Rule of non-simultaneous arrivals to station.

These constraints are the same both for initial scheduling and for rescheduling. All constraints are formulated in meta-notation using previously introduced notation and making possible a simple mapping into an optimization programming language in the implementation phase. For example, we shall consider here in more detail:

Crossing rule

The rule is related to J_p, $J_q \in J$ so that $direction(J_p) \neq direction(J_q)$. 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 $j_p \in \{2,...,k_p-1\}$, $j_q \in \{2,...,k_q-1\}$ and $\mathfrak{Type}(r_l)=st$. From network relation μ it is clear that $\mathfrak{Type}(r_v)=ee$. Hence, $(\overline{d}_{p(j_p+1)} \leq \overline{d}_{q(j_q-1)}) \vee (\overline{d}_{qj_q} + t_c \leq \overline{d}_{p(j_p+1)})$.

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

The rule of no-waiting entering the system

The modeling assumption that each job J_i must be taken for processing at the moment of generation, can be simply expressed by: $\overline{d}_{i1} = d_i$.

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, where 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 a practical importance in operational control and which can be included selectively in the constraint component, depending on requirements put to the rescheduling system. These constraints offer a possibility to have very specific traffic situations planned. The possibilities of defining such constraints are inexhaustible, and our model deals with: Rule of simultaneous stop-over in the station, Rule of time distance between the completion of one and generation of another job, Rule of unavailability of resources, Rule of special separation. For example, we shall consider here in detail:

Rule of simultaneous stop-over in the station

The requests for every two trains to meet in the previously planned station is unsustainable under conditions of the timetable disturbances, and therefore in rescheduling overtaking and crossing stations are determined dynamically. However, 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 $\mathcal{T}ype(r_l)=st$, $\phi_p(j_p)=\phi_q(j_q)=r_l$, for $j_p \in \{1,2,...,k_p-1\}$, $j_q \in \{1,2,...,k_q-1\}$. Let $t_{stop}(cat_i)$ be additional time for stopping train J_i. Then,

$$\min(\overline{d}_{p(j_p+1)}, \overline{d}_{q(j_q+1)}) - \max((\overline{d}_{pj_p} + p_{pj_p} + t_{stop}(cat_p)), (\overline{d}_{qj_q} + p_{qj_q} + t_{stop}(cat_q))) \ge t_m .$$

4.3. Optimization Criteria

The makespan, maximum complete time of all jobs, is the most frequent criterion with the job shop scheduling problems. However, in the train scheduling, the criteria taking into consideration the delays and different priorities and various train categories are of interest. Therefore the following seven relevant optimization criteria, i.e. objective functions have been selected, for which optimization models have been developed:

- minimization of the maximum tardiness $T_{\max} = \max\{T_1, T_2, ..., T_n\}$. Although the criterion minimizes the maximum delay, many trips may suffer disturbances. The criterion is acceptable in situations when passenger trains prevail for scheduling;
- minimization of the maximum weighted tardiness $WT_{\max} = \max\{w_1T_1, w_2T_2, ..., w_nT_n\}$ may be an interesting criterion under the mixed traffic conditions. The weights w_i are usually the same for all trains of the same category within given scheduling period;
- minimization of the total tardiness $D = \sum_{i=1}^{n} T_i$;
- minimization of the total weighted tardiness $WD = \sum_{i=1}^{n} w_i T_i$;
- minimization of the maximum slack of trains in stations, i.e. the minimization of the function $S_{\max} = \max\{\overline{d}_{i(j+1)} - (\overline{d}_{ij} + p_{ij}) \mid 1 \le i \le n,\ 1 \le j \le k_i\ ,\ \mathcal{T}ype(\phi_i(j)) = st\}$. Considering that the absolute compliance with the original timetable corresponds to

the situation that $S_{\max} = 0$, this criterion offers a support to the idea that rescheduled timetable should be as close as possible to the original one;

- minimization of makespan $C_{\max} = \max\{C_1, ..., C_n\}$ expresses our wish for the trains to leave as soon as possible the fragment of railway network under consideration. In the case of rescheduling, this objective function gives support to the localization of disturbances;
- minimization of the number of late jobs $|LJ|$, where $LJ = \{J_i \in J \mid C_i - c_i > 0\}$.

4.4. The choice of strategy, policy and method of rescheduling

Following the ideas presented in [12], the essential steps in implementation of rescheduling are the choice of **factors, strategy, policy and method of rescheduling.**

The rescheduling is activated after recognition of the rescheduling factor. Here the rescheduling factor is a disturbance, i.e. an unplanned forwarding of 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 a general case, the policy may be periodic, event-driven and hybrid. This paper presents a hybrid policy: the rescheduling procedure starts at the end of the defined period if within it some disturbances have arisen.

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

The global rescheduling procedure is represented by pseudo code as follows. In effect, an infinite loop is in question: A disturbance is identified in the first part of the loop body, and its processing is made in the second part.

```
procedure Reschedule (DB, MB)
inputs   DB,          database
         MB,          base of scheduling models
forever
  disturbance ← false
  repeat
    Trigger (DB, disturbance, Active Jobs, $J$, $t_g$)
  until disturbance ∧ $t = k \cdot$period, $k \in Z$
  Select Model (Active Jobs, MB, DB, Model, Max B)
  Select Preparation Model (MB, Model, Preparation Model)
  Prepare ( $J$, PreparationModel, Delays)
  Separate And Schedule Related Jobs ( $J$, Active Jobs, Model, Delays,
  Schedule)
  Save Schedule (DB, Schedule)
  Present Schedule (Schedule)
end forever
end procedure
```

All symbols in this one, but also in the following algorithms 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 concerning the schedule implementation. The occurrence of an unexpected dynamic data in database `DB` triggers a rescheduling procedure. The optimization models available are incorporated in the model base `MB`. This concept offers a support to the idea, based on the user's wish, the period of the day when the rescheduling is made, statistical analysis of the system history, etc., to choose a model that optimizes one or the other optimization function. The procedure ***Select Model***, in accordance with a certain criterion, selects a scheduling `Model` from the model base `MB`. `MaxB` is initial upper bound of objective function. It is the property of a chosen model and expresses the user's wish to reach minimal schedule efficiency. The procedure ***Select Preparation Model*** selects a preparation model corresponding to the chosen scheduling model.

Procedure ***Trigger*** verifies the contents of database `DB`, waiting for information about disturbances. If there is in time moment t a piece of information on unplanned dispatching of one or more trains, the procedure returns a set `ActiveJobs` of all jobs the processing of which is underway at moment t, as well as the set of all jobs J, including the jobs from `ActiveJobs`, but also all not commenced jobs the generation of which is planned up to the given time moment t_g. It is clear that the planning period $[t, t_g]$ should sufficiently exceed the maximum flow time of jobs. Let $J_i \in$ `ActiveJobs` be job dispatched to the station represented by resource r_l, $\mathcal{T}ype(r_l)=st$ and let j-th operation of J_i be performed on that resource, i.e. $\phi_i(j) = r_l$. The rescheduling of the remaining part of the job J_i will be carried out starting from operation o_{ij}, and hence we can assume that the position pos_i from which the job J_i scheduling starts within $[t, t_g]$ is $pos_i = j$. The expected train arrival to that station is a moment of actual generation $\overline{d}_i$ of the rest of job J_i. For jobs $J_i \in J \setminus$`ActiveJobs` $pos_i = 1$, and 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 ***Separate And Schedule Related Jobs*** in the part describing separation heuristics.

The procedure ***Save Schedule*** accommodates the recovered schedule in database, while the procedure ***Present Schedule*** visualizes the schedule in an adequate way.

4.5. Heuristics

Manufacturers of commercial CP tools claim that it is precise enough to formulate what the problem is (the constraint component and objective function), and CP tools are capable to find an optimal solution or a series of feasible solutions thanks to inference incorporated algorithms. Experiments made with ILOG Solver CP tool and its extension for scheduling purposes ILOG Scheduler, 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 requirements. Wishing to take advantage of good properties of CP approach, which have already been discussed, an idea naturally arose to "support" the CP tools by heuristics based on knowledge of the real problem. For this purposes three classes of heuristics have been formulated: bound heuristics, separation heuristics and search heuristics.

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 based on the knowledge of the real problem:

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

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

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

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

2. Lower bound of objective function - is estimated by a special procedure. The estimation 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. 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. Algorithm of procedure ***Prepare*** has the following form:

```
procedure Prepare ( $J$ , PreparationModel, Delays)
inputs              $J$ ,      set of all jobs, includes jobs the processing of which is
                               underway at moment $t$ and non-commenced jobs up
                               to upper bound of planning period $t_g$
   PreparationModel,           scheduling model
returns      Delay,            matrix containing minimum delay if a conflict between two
                               jobs is solved in isolation
forall $J_i \in J$
    forall ( $J_k \in J : i<k$ )
        if not ( $(c_i < d_k) \vee (c_k < d_i)$ ) then
```

```
            origin $\leftarrow$ **min**($d_i, d_k$)
            horizon $\leftarrow$ origin+($c_i - d_i$)+$t_{stop}(cat_i) + t_{start}(cat_i)$ +
            ($c_k - d_k$)+$t_{stop}(cat_k) + t_{start}(cat_k)$
            Jobs $\leftarrow \{J_i, J_k\}$
            LowerBound $\leftarrow 0$
            // setting initial upper bound of objective function
            UpperBound $\leftarrow$ MaxB
            // running of the model solving the conflict between two jobs
            Solve Model (Model, Jobs, origin, horizon, LoverBound,
            UpperBound, Schedule, ObjectiveFunction, Makespan)
            // optimal value of objective function is stored in Delays matrix
            Delays[i,k] $\leftarrow$ ObjectiveFunction
        end if
    end forall
end forall
end procedure
```

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`, based on `Delays` matrix sets a lower bound for the objective function (see next procedure named ***Separate and Schedule Related Jobs***). For objective functions that are not of a sum type, except for the number of late trains, and lower bound has the form:

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

while for sum type objective functions it is of the form :

$$\text{LowerBound} = \mathop{sum}_{J_i, J_j \in RelatedJobs} \{\text{Delays[i,j]} \mid j > i\}.$$

The lower bound of the number of late trains is a number of non-zero rows in `Delays` matrix.

Since the `PreparationModel` 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 include 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 estimation of a lower bound of the 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. Upper bound of objective function – is dynamically bound during the search. For example, if the objective is to minimize total tardiness, then, after finding a feasible solution with total delay D, we add to the model a constraint: $\sum_{J_i \in J} (\bar{d}_{ik_i} + p_{ik_i} - c_i) \le D$.

Initial upper bound of objective function is constant `MaxB`.

The propagation of this constraint reduces the domains of decision variables. In other words, the propagation discards the branches on the search tree that would not lead to better solutions than those already found.

Separation heuristics

The aim is to separate and schedule at the same time only those activities that can influence one another, in order that the system may respond faster to the recognized rescheduling factor.

The set of jobs that must be scheduled jointly is already denoted as `RelatedJobs`. The procedure ***Separate And Schedule Related Jobs*** initially allocates to the set of related jobs a set `ActiveJobs` (the jobs the processing of which is underway at the moment of disturbance). The procedure ***Solve Model*** solves the scheduling model. The set `AdditionalRelatedJobs` includes the jobs that due to cascade effects among operations must be added to the set `RelatedJobs`. The procedure ***Exclude Activities*** has an assignment to recognize in the set `RelatedJobs` those jobs, i.e. their activities that may be considered definitely scheduled and exclude them from further scheduling. The algorithm stops when the set `RelatedJobs` remains empty. This happens in two situations: when all jobs up to the upper bound of the planning period are scheduled, or if a significant time division between jobs that suffer disturbances and the remaining jobs occurred.

```
procedure Separate And Schedule Related Jobs ($J$, ActiveJobs, Model, Delays,
Schedule)
inputs      $J$,          set of all jobs includes jobs the processing of which is underway at
                          the moment of respond to disturbance $t$ and expected non-
                          commenced jobs up to the planning period upper bound $t_g$
ActiveJobs,               the jobs the processing of which is underway at the moment of
                          respond to disturbance
       Model,             selected scheduling model
       Delays,            matrix of minimum delays for solving the conflicts
returns   Schedule,       the schedule of all activities directly or indirectly affected by
                          activities of jobs from the set ActiveJobs
RelatedJobs $\leftarrow$ ActiveJobs
repeat
   origin $\leftarrow \min\{d_i \mid J_i \in$ RelatedJobs $\}$
   horizon $\leftarrow$ origin+ $\sum_{J_i \in RelatedJobs} ((c_i - d_i) + t_{stop}(cat_i) + t_{start}(cat_i))$
   Estimate (Delays, Model, LowerBound)
   UpperBound=MaxB
   // solving the model
   Solve Model (Model, RelatedJobs, origin, horizon, LowerBound,
   UpperBound, Schedule, ObjectiveFunction, Makespan)
   MinGen $\leftarrow t_g$
```

```
    // identification of additional jobs to be generated before maximum complete time of
    // all related jobs and determine minimum moment of such jobs generation
    AdditionalRelatedJobs ← {}
    forall ( $J_i \in J : J_i \notin$ RelatedJobs)
        if $d_i <$ Makespan then
            AdditionalRelatedJobs ← $\{J_i\} \cup$ AdditionalRelatedJobs
            if $d_i <$ MinGen then
                MinGen ← $d_i$
            end if
        end if
    end forall
    // identification of the last station which the train entered not later than the minimum
    // of moments generation of additional jobs MinGen
    forall ( $J_i \in J : J_i \in$ RelatedJobs)
        // the moment of train entering the last station MinGen
        // is a candidate for new generation time of the rest of the $J_i$ ...
        $d_i^n \leftarrow \max_j \{\overline{d}_{ij} \mid \mathcal{Type}(\phi_i(j)) = kol \wedge \overline{d}_{ij} \leq$ MinGen$\}$
        //... the number of operation performed in the last station before MinGen
        // is a candidate for a new position of job generation $J_i$
        $poz_i^n \leftarrow \max\{j \mid \mathcal{Type}(\phi_i(j)) = kol \wedge \overline{d}_{ij} \leq$ MinGen$\}$
    end forall
    // some activities of jobs from the RelatedJobs are definitely
    // scheduled, since they cannot be affected by AdditionalRelatedJobs
    Store Partial Schedule (Schedule, RelatedJobs)
    Exclude Activities (RelatedJobs, AdditionalRelatedJobs, MinGen)
    RelatedJobs ← RelatedJobs $\cup$ AdditionalRelatedJobs
// the algorithm stops when the set RelatedJobs becomes empty
until RelatedJobs = {}
end procedure
```

Search heuristics

The aim is to allocate start times to activities as early as possible so that all imposed constraints are satisfied, i.e. find a feasible solution.

The procedure ***Set Start Times Of Activities*** finds start times of activities such that they define a feasible solution. The order of variables is in accordance with increasing lower bound of their domains, as well as the order of values within a domain. As soon as a start time is assigned to an activity, the domains (intervals) are updated corresponding to start times of unscheduled activities. At the moment one of the domains remains empty, by backtracking the algorithm tries to find a node in search where a wrong decision has been taken. The algorithm stops when all activities have obtained the start times (i.e. a feasible solution has been found), or if there is no alternative after an error (no solution).

```
procedure Set Start Times Of Activities (Jobs)
inputs Jobs,     set of jobs to be scheduled starting from their actual positions $pos_i$
returns  $\{\bar{d}_{ij} \mid J_i \in$ Jobs, $pos_i \le j \le k_i\}$,
                        set of start times for all activities of jobs starting from
                        their actual position $pos_i$
begin
   Activities $\leftarrow \{o_{ij} \mid J_i \in$ Jobs, $pos_i \le j \le k_i\}$
   PostponedActivities $\leftarrow \{\}$
   ScheduledActivities $\leftarrow \{\}$
   $d_0 \leftarrow \min\{d_{ij} \mid o_{ij} \in$ Activities $\}$
   repeat
      SimultaneousActivities $\leftarrow \{o_{ij} \mid o_{ij} \in$ Activities $\wedge d_{ij} = d_0\}$
      k $\leftarrow$ Cardinality (SimultaneousActivities)
      // creating array of all subsets of SimultaneousActivities without empty
      // set, arranged by non-decreasing cardinality
      Create Array Of Subsets (SimultaneousActivities, ArrayOfSubsets)
      n $\leftarrow 2^k - 1$
      ok $\leftarrow$ false
      repeat
         PostponedActivities $\leftarrow \{\}$
         // test if the set of operations incorporated in ArrayOfSubsets[n] can
         // start at moment $d_0$
         if Can Start (ArrayOfSubsets[n], $d_0$) then
            PostponedActivities $\leftarrow$ SimultaneousActivities \
            ArrayOfSubsets[n]
            Update Start Times (PostponedActivities)
            // a new minimum start time for remaining activities
            $d_a \leftarrow \min\{d_{ij} \mid o_{ij} \in$ Activities\SimultaneousActivities$\}$
            $d_p \leftarrow \max\{d_{ij} \mid o_{ij} \in$ PostponedActivities $\}$
            if not $(d_p < d_a)$ then
               ScheduledActivities $\leftarrow$ ScheduledActivities $\cup$
               ArrayOfSubsets[n]
               ok $\leftarrow$ true
            else
               n $\leftarrow$ n-1
               // backtracking to select another set of activities to start at moment $d_0$
               Backtrack
            end if
            **end if**
        **until** ok
        $d_0 \leftarrow d_a$
    **until** ScheduledActivities = Activities
**end procedure**
```

The procedure ***Solve Model*** calls procedure ***Set Start Times Of Activities***, and thereafter, based on a found feasible solution a new upper bound for the objective function is set, and propagation of the corresponding additional constraint reduces the domains of other decision variables. The procedure ***Solve Model*** stops further search if the upper bound of the objective function matches up with the lower bound or if rescheduling of start times of activities does not lead to a better value of the objective function.

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

4.6. Surrogate objective function

Based on the nature of CP approach it is clear that the optimization of a small domain function is faster than the optimization of a larger domain function, except if in the latter case there is a very powerful heuristic. It is a reasonable idea, therefore, to use, instead of the real objective function with a large domain a surrogate objective function with a smaller domain, if there is such. We can minimize a surrogate objective function, but with an additional dynamic constraint, such that minimization may go only through those feasible solutions that do not enhance the actual objective function. For example, let makespan is the actual objective function and $|LJ|$ is its surrogate objective function. The values in domain of $|LJ|$ should be in the descending order; it is reasonable, because we expect "good" makespan if several trains suffer small disturbances. After every assigning the value to $|LJ|$, the constraint $\max_{J_i \in J}(\overline{d}_{ik_i} + p_{ik_i}) \leq \text{makespan}$ will be activated. Hence, if the estimated number of conflicts exceeds a threshold, procedure ***Select Model*** should select a model with the surrogate objective function, if there is such, from the modelbase MB.

5. METHOD EVALUATION

Validation of heuristic algorithms is in a general case very complicated. One of the ways is their experimental verification. For this purpose, the first prototype of software system for train rescheduling has been designed and implemented.

CP tool ILOG Solver and its upgrade for scheduling purpose ILOG Scheduler, manufactured by French company ILOG (http://www.ilog.com), have been used in implementation. ILOG Scheduler permits to create models in the terms of resources, activities and time constraints. The optimization models in our case are formed in OPL modeling language, while the combination and control with optimization models has been achieved by using a procedural language OPL Script. The integrated development environment OPL Studio enabled us to create and modify the models using OPL, to combine and manage the models using the language OPL Script, and to run the models by ILOG Solver and ILOG Scheduler. The trial version of OPL Studio is available on Internet (http://www.ilog.com/download/opl) and this has been actually used for implementation of the first prototype of train scheduling system.

Figure 3 presents a realized window of user interface of this system – a graph presentation of the recovered train schedule for infeasible schedule shown in Figure 2.

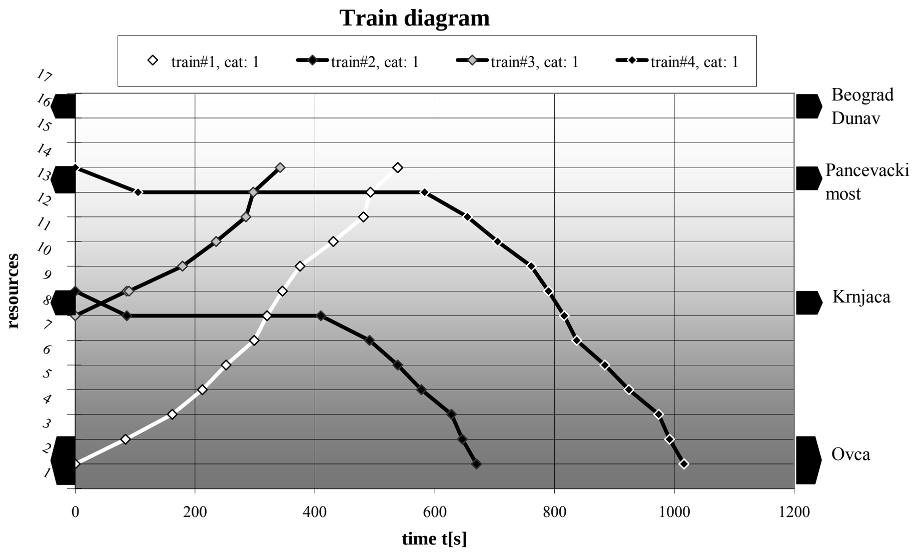

Figure 3: Recovered timetable for infeasible timetable in Figure 2 if the objective is the minimization of total tardiness D

Experiments have been carried out on a fragment of real railway network (a part of Belgrade Railway Junction), with actual train categories operating there, but **with traffic frequency immensely exceeding the real one**. The jobs (trains) are "piled up" on purpose to test the endurance of the method. The train categories were joined by priorities assessed by expertise. All seven relevant objective functions, discussed in 4.3, participated in the experiments. Table 1 in Appendix A presents a yield of heuristic algorithms on selected examples that differ with respect to numbers of jobs for rescheduling and initial numbers of conflicts. Each set of jobs suffering disturbances includes trains of different categories and different movement directions. All experiments have been implemented on personal computer Intel (R) Pentium(R) 4 CPU, 2GHz. From the analysis of experiment results the following conclusions may be drawn:

- CPU time of schedule recovery depends on the number of activities and number of conflicts;
- solving initial conflicts may bring up additional conflicts;
- in most cases the time performance and solution quality is satisfactory;

- a 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);
- the approach of "formal" minimization of the function, and dynamic limiting of the value of the other function in 90% of tested cases proved to be extremely efficient: an arrangement with the same value of the actual objective function is found, the run time of the model is reduced by around three times! If actual objective function is makespan and surrogate objective function is $|LJ|$, run time is 21.59 instead of 58.88 for the sixth example form Table 1.

6. CONCLUSION

The paper presents a very realistic railway transport model. Namely, actual line-side signals that limit resources have been taken into account. These signals are used for control if a train may proceed its trip on a particular resource. In the available literature [7] the authors manipulate with approximate time spacing and not with real spatial spacing of trains, which may be notably different if there is a significant difference in length of resources and in movement speeds of trains. Also, as opposed to [10] the traffic mixture has been taken into account, so that different priorities have been allocated to different train categories.

The train rescheduling problem has been formulated and solved as constraint satisfaction optimization problem. Corresponding optimization criteria take into consideration delays and established priorities between different train categories (e.g. the maximum tardiness, total tardiness, maximum weighted tardiness, etc.). In order to improve the time performance of available constraint programming tool ILOG Solver and to meet the rescheduling requirements, three classes of heuristics working together on seeking the solution have been proposed. They are referred to as separation heuristic, bound heuristic and search heuristic. In special cases, instead of the actual objective function with a large domain, a surrogate objective function has been used with a smaller domain.

For the purpose of an experimental verification of proposed heuristic algorithms, the first prototype of software tool 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 Junction), with real train categories in operation in that junction and with real possible disturbances, have proven a validity of the described approach, both in time performance and in solution quality.

Although the main objective of the research is an operational reconstruction of the timetable under the conditions of disturbances, the described approach has a high degree of universality within the given problem category. By a relaxation of strict time limits and an increased size of the problem, the method is capable to solve the problems of initial train scheduling on a real network up to optimization within reasonable time.

Also, by solving a difficult problem of train rescheduling, we have paved the way for solving a whole series of problems, the core of which is the train scheduling, e.g. timetable preparation, determining of economically acceptable capacity utilization interval, the estimation of train stopping and waiting for traffic reasons, anticipation of the results of investment activities, identification of bottlenecks in infrastructure, choice of possible solutions of conflict points, research on allocation of block sections and line-side signals etc.

The research presented in this paper should not be considered as a closed system. It might be of interest to further upgrade the network model, a constraint component and special constraints, in particular, as well as the described heuristic algorithms. Also, another research could be made about criteria for choosing a surrogate objective function for the given actual one.

REFERENCES

[1] Bater, W.M., "Computer aided railway engineering", in: Mellit, B., Hill R.J., Allan J., Sciutto, G., Brebbia, C.A. (Eds.), *Computers in Railways VI*, WIT press - Computational Mechanics Publications, Comreco Rail Ltd York, England, 1998, 199-211.

[2] Cai, X., and Goh, C.J., "A fast heuristic for the train scheduling problem", *Computers and Operations Research*, 21(5) (1994) 499-510.

[3] Chiu, C.K., Chou, C.M., Lee, J.H.M., Leung, H.F., and Leung, Y.W., "A constraint-based interactive train rescheduling tool", *Proceedings of Second International Conference on Principles and Practice of Constraint Programming*, 1996, 104-118.

[4] Cowling, P., Johansson, M., "Using real time information for effective dynamic scheduling", *European Journal of Operational Research*, 139(2) (2002) 230-244.

[5] Čicak, M., Vesković, S., and Mladenović, S., *Models for Establishing the Railway Capacity*, Faculty of Transport and Traffic Engineering and Želnid, Belgrade, 2002. (in Serbian)

[6] Jones, A., and Rabelo, L.C., "Survey of job shop scheduling techniques", Technical Paper, NISTIR, National Institute of Standards and Technology, Gaithersburg, MD, 1998. (Downloadable from website http://www.mel.nist.gov/msidlibrary/doc/luis.pdf)

[7] Kreuger, P., Carlsson, M., Olsson, J., Sjoland, T., and Astrom, E., "Trip scheduling on single track networks – the TUFF train scheduler", Workshop on Industrial Constraint Directed Scheduling, 1997, 1-12. (Downloadable from website http://citeseer.ist.psu.edu/cache/papers/cs/2088/http:zSzzSzwww.sics.sezSz~alfzSzcp_97_ws.pdf/kreuger97trip.pdf)

[8] Marriott, K., and Stuckey, P.J., *Programming with Constraints*: *An Introduction*, The Massachusetts Institute of technology Press, Cambridge, 1998.

[9] Mladenović, S., Vesković, S., and Čicak, M., "SIZES – software for establishing the capacity of the single track", *Proceedings of XLV ETRAN Conference*, Bukovička Banja, Volume III, 2001, 63-66. (in Serbian)

[10] Oliveira, E., and Smith, B.M., "A hybrid constraint-based method for single-track railway scheduling problem", Report 2001.04, School of Computing, University of Leeds, 2001. (Downloadable from http://www.comp.leeds.ac.uk/research/pubs/reports/2001/2001_04.pdf).

[11] Pinedo, M., *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, 1995.

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

APENDIX A

Table 1: A yield of heuristic algorithms in the schedule recovery - selected real examples

No. of example	**Initial no. of conflicts**	**No. of jobs involved in the distur-bance**	SCHEDULE PERFORMANCES								
			No. of conflicts solved	**CPU time**	$T_{\max}$	$WT_{\max}$	D	WD	$S_{\max}$	**make-span**	$\lvert LJ \rvert$
1.	1	2	1	3.84	250	500	370	980	250	2634	2
			1	3.38	250	500	370	980	250	2634	2
			1	4.09	250	500	370	980	250	2634	2
			1	4.24	250	500	370	980	250	2634	2
			1	20.44	250	500	370	980	223	2634	2
			1	3.72	250	500	370	980	250	2634	2
			1	3.30	440	1760	440	1760	320	2766	1
2.	2	3	2	9.54	450	900	809	1858	450	6681	3
			2	9.15	450	900	809	1858	450	6681	3
			2	9.84	576	2304	884	2920	456	6499	3
			2	9.32	484	968	913	2066	309	6751	3
			2	8.97	484	968	913	2066	309	6751	3
			2	9.17	576	2304	884	2920	456	6499	3
			2	7.59	992	1984	1112	2464	992	7156	2
3.	3	3	3	10.74	641	2564	1064	3015	521	2067	3
			3	10.18	794	1632	1230	2482	428	2193	3
			3	9.69	641	2564	1064	3015	521	2067	3
			3	10.26	794	1632	1230	2482	428	2193	3
			3	10.90	794	1632	1230	2482	428	2193	3
			3	9.88	641	2564	1064	3015	521	2067	3
			5	8.62	641	2564	1064	3015	521	2067	3
4.	3	5	3	12.59	532	1064	1237	2296	532	7330	5
			3	13.29	532	1064	1237	2296	532	7330	5
			3	16.83	532	1064	1237	2296	532	7330	5
			3	17.83	532	1064	1237	2296	532	7330	5
			3	28.64	532	1064	1478	2537	363	7330	5
			3	14.67	854	1708	1331	2484	854	7102	4
			5	21.59	1904	3808	3556	7112	1652	7816	2
5.	5	5	6	25.77	918	3672	2463	7578	392	2854	4
			6	23.48	920	3024	2580	7536	578	2878	4
			6	27.78	918	3672	2457	6990	392	2754	4
			6	28.24	918	3672	2457	6990	392	2754	4
			6	24.33	918	3672	2463	7578	392	2854	4
			6	30.73	918	3672	2457	6990	392	2754	4
			6	24.12	918	3672	2463	7578	392	2854	4
6.	6	7	8	24.54	622	1244	2727	5351	619	6867	7
			8	18.82	622	1244	2727	5351	619	6867	7
			8	32.36	673	1346	2484	4677	673	7055	6
			8	32.13	673	1346	2484	4677	673	7055	6
			8	53.99	622	1244	2727	5351	619	6867	7
			8	58.88	622	1244	2727	5351	619	6867	7
			8	46.55	673	1346	2484	4677	673	7055	6