Reflective Nested Simulations Supporting Optimizations within Sequential Railway Traffic Simulators

ROMAN DIVIŠ and ANTONÍN KAVIČKA, Faculty of Electrical Engineering and Informatics, University of Pardubice, Pardubice, Czech Republic

This article describes and discusses railway-traffic simulators that use reflective nested simulations. Such simulations support optimizations (decision-making) with a focus on the selection of the most suitable solution where selected types of traffic problems are present.

This approach allows suspension of the ongoing main simulation at a given moment and, by using supportive nested simulations (working with an appropriate lookahead), assessment of the different acceptable solution variants for the problem encountered—that is, a what-if analysis is carried out. The variant that provides the best predicted operational results (based on a specific criterion) is then selected for continuing the suspended main simulation. The proposed procedures are associated, in particular, with the use of sequential simulators specifically developed for railway traffic simulations. Special attention is paid to parallel computations of replications both of the main simulation and of supportive nested simulations.

The concept proposed, applicable to railway traffic modelling, has the following advantages. First, the solution variants for the existing traffic situation are analyzed with respect to the feasibility of direct monitoring and evaluation of the natural traffic indicators or the appropriate (multi-criterial) function. The indicator values compare the results obtained from the variants being tested. Second, the supporting nested simulations, which potentially use additional hierarchic nesting, can also include future occurrences of random effects (such as train delay), thereby enabling us to realistically assess future traffic in stochastic conditions.

The guidelines presented (for exploiting nested simulations within application projects with time constraints) are illustrated on a simulation case study focusing on traffic assessment related to the track infrastructure of a passenger railway station. Nested simulations support decisions linked with dynamic assignments of platform tracks to delayed trains.

The use of reflective nested simulations is appropriate particularly in situations in which a reasonable number of admissible variants are to be analyzed within decision-making problem solution. This method is applicable especially to the support of medium-term (tactical) and long-term (strategic) planning. Because of rather high computational and time demands, nested simulations are not recommended for solving short-term (operative) planning/control problems.

CCS Concepts: • **Computing methodologies → Agent / discrete models**;

Additional Key Words and Phrases: Railway traffic simulation, nested/recursive simulations, automated decision-making support, parallel discrete event simulation

The research presented in this article was supported by the ERDF/ESF project Cooperation in Applied Research between the University of Pardubice and companies in the Field of Positioning, Detection, and Simulation Technology for Transport Systems (PosiTrans)—No. CZ.02.1.01/0.0/0.0/17_049/0008394.

Authors' addresses: R. Diviš and A. Kavička, Faculty of Electrical Engineering and Informatics, University of Pardubice, Studentska 95, 532 10, Pardubice, Czech Republic; emails: {Roman.Divis, Antonin.Kavicka}@upce.cz.

1049-3301/2021/09-ART1
https://doi.org/10.1145/3467965

ACM Reference format:
Roman Diviš and Antonín Kavička. 2021. Reflective Nested Simulations Supporting Optimizations within Sequential Railway Traffic Simulators. *ACM Trans. Model. Comput. Simul.* 32, 1, Article 1 (September 2021), 34 pages.
https://doi.org/10.1145/3467965

1 INTRODUCTION

The use of computer simulation methods when analyzing railway system problems is convenient in that it enables the properties and behavior of the system to be flexibly tested in different traffic conditions. The objectives of studies examining railway traffic systems are linked with, for instance, providing supports when setting up or optimizing timetables, assessment of the capacity of railway lines and stations, testing alternative approaches to traffic control, and analysis of the effects of train delays and other traffic disruptions. Computer simulation can be used to quite efficiently test different timetables, technological processes, or different rail yard infrastructure topologies. Examination in the reality domain would be extremely costly and time-consuming, while computer simulations enable us to rapidly and efficiently examine the feasible variants of the system analyzed without having to perform actual physical changes in the systems. Many well-established simulation tools specializing in the examination of rail traffic are currently available. The tools, some of which are mentioned below, are implemented as sequential simulators and are typically run on PCs.

Simulation models mirroring railway traffic can use different degrees of abstraction depending on the nature of examination to be performed. Models working on the microscopic level of detail (microscopic simulators) enable each railway process (such as partial train shunting, the change times of passengers between connecting trains, etc.) to be examined in detail. Such models include detailed information on rail yard infrastructure, interlocking systems, control and technological processes, trains and individual train cars, service staff, and sometimes the passengers. Microscopic simulators are typically used to examine traffic in specific railway network segments, which may include both railway lines and railway nodes [1–3].

Mesoscopic simulators [4, 5] typically use simplified models of technological processes and are normally used for assessing the total throughput/capacities of the lines/stations and/or for examining the traffic quality (determined mainly by the train delay characteristics within the lines/stations analyzed). Such simulators, however, do not abstract from the objects of the trains whose life cycles are examined within the simulation experiments.

Macroscopic simulators [6], on the other hand, are typically focused on railway traffic in geographically wide areas: rather than the individual train entities, it is the changing coarse traffic characteristics (such as the extent of line/station use, mean train speed in different line segments, etc.) that are examined on a highly simplified infrastructure.

Railway traffic simulations can work both in the *deterministic mode* and in the *stochastic mode*. The former is typically applied for primary examination of whether a timetable is or is not non-conflict. Hence, the trains are assumed to run precisely according to the timetable (no delays are applied) in the simulation. The goal is to determine whether the train motions adversely affect one another (i.e., whether any train competition for the same part(s) of the track infrastructure results in train delays). This deterministic investigation is typically followed by simulation examination of the traffic with the inclusion of random effects, such as train delays or train/technical facility failures/faults/defects. Stochastic simulations, in which the planned traffic is disturbed, include conflict situations that must be addressed. The request by a delayed train for allocation of a part of the track infrastructure (platform/line track, railway route, etc.) that is currently occupied by another train is an example of a conflict.

The way a conflict is resolved during a simulation experiment can be based on one of two different approaches. In the *interactive approach*, the simulation is suspended and how the conflict is to be resolved is decided by the simulator user (expert). The simulation is then resumed using the user's method of conflict resolution. In the *software support* approach, the decision-making procedure is computer aided, that is, the simulation model selects the solution automatically, without the need to suspend the simulation calculation. An overview of the methods that are used to support decision-making within railway traffic simulators is presented in the next section.

The objective of the R&D activities, the results of which are summarized here, was to design and test a novel automated approach to support optimizations/decision-making concerning the resolution of traffic conflicts within sequential rail-traffic simulations. The approach to the problem is based on the concept of hierarchically nested simulations. The method is first theoretically described, a technique for parallel executing of nested simulations (pertaining to different traffic variants on different hierarchic levels) is devised, and the guidelines for exploiting nested simulations (within application projects) are demonstrated on a case study.

This research was motivated by efforts to extend the range of optimization methods applicable within sequential rail traffic simulators. Such simulators are currently quite widely used in the rail traffic application domain specifically to examine the available traffic (timetable) variants and traffic infrastructure.

2 OVERVIEW OF DECISION SUPPORTS WITHIN RAILWAY-TRAFFIC SIMULATORS

Railway traffic simulators can address emerging traffic conflicts during simulation experiments by applying different approaches/methods. The solution can be obtained either directly from the simulator user (interactive simulation mode) or it can emerge from automated computation based on exact or heuristic methods of *operations research* (such as *integer linear programming, multicriterial evaluation of variants*), *soft computing methods* (e.g., using *artificial neural networks, fuzzy logic*, etc.), expert static *priority planning*, and so on. Another option is the use of what is called *reflective nested simulations*, evaluating predictively the different solution variants for a given lookahead (with respect to the moment of occurrence of the conflict).

2.1 User-Based Decision-making

If the automatic simulation process is suspended and switched to the interactive mode when a conflict is encountered, it is the user who makes the decision regarding conflict resolution. This solution is then used in the resumed automatic computation, which is suspended again when the next conflict is encountered. This approach may be satisfactory for some experiments and can then be applied in selected training simulator types that do not serve real-time training. Such training simulators enable the users to be trained in various controller areas; they use running simulations that are steered by the user solutions input to them. The product Villon is an example of a commercial railway traffic simulator that can be used for this purpose [1].

If, however, time-consuming simulations with a large number of replications are required, the number of conflict states increases considerably. In this case, simulation with interactive decision-making would be slow and difficult for the user. Also, user decision-making is typically not optimal in such situations and may result in a poorer quality of simulation results. Thus, it is more appropriate to use automated software-based decision-making support in such situations.

2.2 Static Priority Planning

Railway traffic conflicts typically emerge from train competition for resources (e.g., parts of the track infrastructure, service staff, technical facilities, shunting engines, etc.). A conflict at a given

moment is typically represented by a request for allocation of a resource that is not available at that moment. Railway traffic requires the existence of appropriate (static) traffic plans such as timetables, railway station track occupation plans, or service staff working schedules based on which traffic is organized. Conflicts emerge in situations in which none of the plans can be adhered to and an alternative solution must be sought.

Static *priority planning* is a simple method of automated decision-making regarding conflicts, which uses dedicated commercial software tools (OpenTrack [3], Villon [1]) for railway traffic simulations. A priority queue of feasible alternative solutions (regarding the allocation of resources) is set up for each conflict type before launching the simulation. Allocation of a resource in accordance with the operational plan is normally the highest-priority solution. The highest-priority usable solution is automatically selected when a conflict is encountered in the simulation. If none of the solutions in the queue is feasible, the entity requesting allocation of the resource normally waits until any of the alternative solutions is feasible.

The following are examples of priority queues: (i) list of alternative platform tracks for the dwell of passenger trains arriving to the railway station from direction D_i and departing in direction D_j; and (ii) list of alternative tracks for transit freight trains arriving at the station from direction D_k and departing in direction D_l. Priority queues should preferably be set up by railway traffic experts who are able to competently decide which alternative solutions are applicable. In this respect, no precisely specified optimization criterion is formally applied. Instead, this is based on the expertise of the professional who assesses the extent of disturbance of the planned operation for each alternative solution. It is noteworthy that preparation of the queues for extensive modelled systems is a rather laborious task.

Still, priority planning based on expertly compiled priority queues does not enable detailed automated evaluation in order to decide which solution is objectively better than the other ones or what impacts the solutions may have on future traffic.

2.3 Expert Systems

Expert systems constitute a class of computer programs aimed at providing expert advice in line with the expertise of professionals in the application domain in question. Expert systems are categorized into basic classes based on the types of use. Typical representatives include software systems executing various diagnostics or planning tasks. Diagnostic expert systems serve to find the diagnosis (solution) of a given problem. This system type can be applied not only in medicine to identify a disease in a patient but also, for instance, in the transport area to solve traffic logistic problems and other problems [7].

In contrast to conventional programs, the expertise of the expert systems is stored separately from the data. The inference mechanism controlled by the expert system uses both the data and the expertise to find the solution. The inference mechanism itself can be based on different principles (e.g., evaluation of logic rules, fuzzy logic, artificial neural networks).

Expert systems are also used in the railway transport domain, where they provide a means for operative traffic control and train movement and train interaction planning. Fay and Schnieder [8] described the basic issues of railway traffic control and planning using expert systems and presented an overview of existing expert systems used in this area. They also presented their own proposal for an expert system using expertise and fuzzy logic. The system was improved as described in [9], devising and describing a supporting control system based on fuzzy Petri nets.

ESTRAC-III is another example of an expert system based on rules and used for railway traffic planning [10]. The output is a plan of changes that is applied to the existing graphic railway timetable. The decision-making system also uses computer simulations to identify the impacts of train delays and puts forth planned measures. The simulation is based on the "partial simulation"

concept in which interrelated events associated with the various stations are grouped together, thereby dividing the simulated railway traffic system into partial subsystems [11].

The Sepia real-time expert system addresses a similar problem [12]. Unlike the system described earlier, the decision-making mechanism of this system is based on searching through and modifying the structure of the state graph.

The system for freight transport on the Daqin Line, China [13], which is based on fuzzy logic, is an example of an expert system currently applicable to operative railway traffic control. An expert system based on fuzzy Petri nets has been devised for addressing transport outage situations and has been tested within a case study of the railway system in Taiwan [14].

Although these expert systems are usually designed for application in real environments, they can also be used as a decision-making supporting tool in simulators [15]. Despite this, those expert systems are not yet ready for use in general railway traffic simulators.

2.4 Methods of Operations Research and Soft Computing

Operations research is an extensive field of applied science encompassing a wide range of exact and heuristic optimization methods, some of which may potentially be usable for automated decision-making support within railway traffic simulators. Examples include mixed integer linear programming methods [16, 17], multicriterial decision-making methods [18], graph theory methods [19], and the like.

Some soft computing methods/approaches are also potentially usable for obtaining solutions regarding conflict states in traffic simulations. Artificial neural networks [20], fuzzy Petri nets [14] and more can be applied in the decision-making components of the simulators.

These methods/approaches are typically tested within case studies but are normally not included in commonly available commercial railway traffic simulators, largely because it is rather difficult within the simulators to modify or parameterize the methods (particularly because of specification of the required optimization criterion) in a user-friendly manner for addressing the given types of conflicts.

2.5 Nested Simulations

Nested simulations, sometimes also referred to as *recursive simulations*, constitute another method applicable to decision-making support in simulators.

Sequential Simulators

The concept of the *reflective nested simulation* (*RNS*) [21] consists of suspending the *main sequential simulation* if a *conflict* (*decision point*) occurs, and cloning it into variants. The clones (supporting RNS) are differently parameterized in order to test the various conflict resolution options (*what-if analysis*) in the suspended main simulation. These recursive sequential simulations (actually representing different lookaheads) are launched, stopped in a certain limited time and evaluated. The variant that provides the best predicted results (according to a defined criterion) is then used for continuation of the suspended main simulation.

RNS, which can find use in different application domains, has been investigated by a number of scientists. E. Kindler, who published several papers devoted to this topic, focused on a theoretical description of nested simulations [22], their categorization, related terminology [21], and practical applications in various domains [23].

Gilmer and Sullivan examined the effectiveness of a large number of replications against what is referred to as *multitrajectory simulation* [24]. This type of simulation is an alternative way to examine the state space. The various simulations are branched when conflicts occur, and new

simulations emerge in order to test a considerable number of different alternatives of the course of stochastic simulations. The authors concentrated mainly on the Eaglet military simulator, which simulates the movement of the troops of two armies and their interactions. They also examined the topic of RNS use for supporting the decision-making process [25].

Furthermore, nested simulations can be used to set up a scalable simulation. The nested simulation approach is used there for the calculation of transformations to pass from submodels on a macroscopic level to submodels on a microscopic level [26]. A specific group of publications describe the use of nested simulations (two-level simulations) for the needs of financial and risk management [27].

Parallel Simulators

Many authors also studied the topic of *cloning parallel discrete event simulations* (*PDES*) [28–30]. In contrast to sequential simulations, PDES use parallel computations (on multiple computation nodes) for the *logical processes* belonging to the simulating system in question. The *cloning method* is used with respect to certain *decision points* for subsequent parallel testing of different evolution variants of the simulation experiment. When using the cloning method, specific attention is paid to the optimization of the computation process, especially in the context of the feasibility of (i) some computations being shared by multiple parallel processes, (ii) eliminating (pruning) non-promising computation branches, and (iii) saving snapshots of selected parts of the simulator's state space in an external memory on an ongoing basis for their later potential use (e.g., for *rollbacks* when performing parallel and distributed simulations applying the *optimistic synchronization method*).

Specifically, the authors of [28] present the concept of *virtual logical processes*, enabling one to avoid repetition of computations that are common to multiple clones, thereby helping improve the overall effectiveness of the complex parallel computations of the simulation experiments.

The authors of [29] introduce the *agent-based bottom-up cloning strategy* (using an *incremental cloning mechanism*), in which *cloning trees* are heuristically generated. Such trees support the identification of the potential existing in computation sharing among different clones with similar parametrization, thereby promoting the efficiency of the parallel simulations. This strategy was illustrated on the application of the computations to a *sequential CPU platform*, a *multi-core CPU platform* with *OpenMP* enabled, and a *GPU platform*.

The approach introduced in [30] is based on the application of *granular cloning*. This concept uses *versioned objects* that share their states for a certain period of the simulation until a specific state emerges. Then, they start to deviate from sharing and instead develop through state trajectories specific for each version of the simulation scenarios. The scenarios are not a priori explicitly specified in this approach, but a log must be maintained during the simulation in order to preserve the dependence relationships among the objects. Granular cloning brings about savings in the computations, particularly where the state trajectories of the different scenarios are similar.

The authors of [31] present the mechanism of a manager for preserving the logical process states in distributed simulators (applying the *optimistic synchronization method*). This manager is referred to as an *Autonomic State Manager* (*ASM*), which does not require the application programmer to provide it with a serialization/deserialization module for taking snapshots of the logical process states. This support is suitable for distributed simulation programmers—the ASM implements solutions for both *incremental* and *non-incremental log modes*, from which a suitable variant is selected depending on the development of the optimistic distributed simulation dynamics.

The focus of [32] is on the description of an original *distributed middleware* that enables PDES applications, initially programmed for systems using shared memory, to calculate also on *clusters*

of (cloud) resources. This middleware uses a specific synchronization protocol that enables *cross-simulation object access* through appropriate event handlers. Implementation solutions that were experimentally verified exhibited fairly good computation effectiveness. This solution supports very well the execution of PDES applications on clusters, partly relieving the developers of the task of coding interactions between the partial models of the parallel/distributed simulator.

The authors of [33] focus on the issue of simulation cloning, in which the application of the paired batches of clones is utilized. A relevant pair is composed of a basic batch and a corresponding counterpart batch. The basic batch contains primary replications of the relevant simulation. The counterpart batch utilizes random variables with *induced negative correlation* against the runs of primary replications from the basic batch. The purpose of applying that approach is to achieve a variance reduction in the statistical processing of the output data. Particular attention is paid to determining the *optimal number of clones* that are initiated on the occurrence of decision points. Relevant optimizations are based on maximizing the *efficiency* of the simulation. That efficiency is defined to be the reciprocal of the product of the variance and computational costs per replication. Due to the need to perform optimizations for each decision point, this approach is more suitable for applications in which there are no massive random occurrences of decision points.

Augmentation of Sequential Simulators with RNS-Based Techniques
This article is mainly devoted to the possibilities of augmenting existing sequential simulators with RNS-based support to the decision-making processes. The simulators are typically used for experiments associated with large numbers of replications in order to get statistically significant results. Based on the above facts, the following can be stated:

- The basic principle of RNS implementation is relatively simple, and the possibility of using the simulation engines from existing sequential simulators to run them is assumed. A detailed description of nested simulations applied within sequential simulators and their exploitations within application projects with time constraints are presented in the sections that follow.
- Approaches aimed to improve the effectiveness of computations (using PDES) are applicable in the domain of sequential simulations to a limited extent because sequential simulators consist of a single logical process and, hence, unlike parallel simulators, are not faced with the challenge of applying complex solutions to storing the simulation states or of using complex techniques of computation sharing among many different logical processes (from distinct clones). Thus, the focus is mainly on appropriate solutions for parallel computations of a potentially large number of replications.

3 CONFLICT RESOLUTION USING REFLECTIVE NESTED SIMULATIONS

As mentioned earlier, reflective nested simulations constitute one of the feasible approaches in support of conflict resolution (e.g., in railway traffic simulators). The text that follows focuses both on the technical aspects of the RNS and on the concept of RNS deployment in practical applications.

3.1 Primary Technique of RNS Utilized for What-If Analysis

Implementation of software support for automated conflict resolution (applying RNS) during simulation experiments can be based on the use of a standard simulation engine (implemented within the sequential simulator being used) both for the main simulation and for the nested simulations. Nevertheless, the original control procedure [34] is used for the integrated computation that includes nested simulations.

Before introducing the computation concept, let us describe and explain some symbols, parameters, functions, and software components (specified in Table 1) whose setting/implementation affects the evolution of the simulation experiments.

The main part of the control algorithm related to the *r*-th replication of the main simulation using RNS (i.e., *maxLevel* > 0) is as follows:

ALGORITHM 1: Control of the r-th Replication

Control of the r-th replication representing the main simulation

01 The conflict counter is initialized: i ← 1.

02 The execution of the *r*-th replication of the main simulation (${}^{d}S_{0,0,r}$) is launched.

03 ${}^{d}S_{0,0,r}$ computation is run until the nearest (*i*-th) conflict is encountered at moment t_i of the simulation time, at which the simulation is suspended.

04 *VariantsGen(i)* is used to propose *n* alternative variants for resolution of the *i*-th conflict, where $n \leq MaxVariants$.

05 The state space of ${}^{d}S_{0,0,r}$ is cloned and for each resolution variant of the current (*i*-th) conflict, the corresponding nested replications (${}^{d}S_{i,j,k,}$ $j = 1 \ldots n$, $k = 1 \ldots nestReplCount$) are initialized.

06 The nested replications are completely calculated: ${}^{d}S_{i,j,k}$, $j = 1 \ldots n$, $k = 1 \ldots nestReplCount$. The computation of each replication is terminated when the terminating condition *stopCond* is met for it.

07 The results for the *n* different variants of the *i*-th conflict resolution are evaluated. The evaluation of each *j*-th variant ($j = 1 \ldots n$) is based on a statistical evaluation of the results of its *k* replications ($k = 1 \ldots nestReplCount$). Subsequently, that variant of resolution of the *i*-th conflict that provided the best result in terms of the function *CrOptim(j)* is selected for continuation of ${}^{d}S_{0,0,r}$.

08 The conflict counter is updated: $i \leftarrow i + 1$.

09 Return to step 03.

This algorithm is illustrated in Figure 1(b), with a schematic picture of the hierarchically ordered set ${}^{1}H_3({}^{1}S_{0,0,r})$ consisting of 22 completely computed replications:

$$
\begin{aligned}
{}^{1}H_3({}^{1}S_{0,0,r}) = \{{}^{1}S_{0,0,r}\} \cup \{{}^{1}S_{i,j,k} \mid & i = 1; j = 1 \ldots 3;\ k = 1 \ldots 3\} \\
\cup \{{}^{1}S_{i,j,k} \mid & i = 5, 7;\ j = 1 \ldots 2;\ k = 1 \ldots 3\}.
\end{aligned}
$$

The highest (zeroth) hierarchic level is represented by the *r*-th replication of the main simulation (${}^{1}S_{0,0,r}$); the first level contains the nested simulations (*maxLevel* = 1).

The conflicts ${}^{1}C_1$, ${}^{1}C_5$, ${}^{1}C_7 \in Confl({}^{1}S_{0,0,r})$ that occurred during the main simulation are resolved by using RNS. Three replications (*nestReplCount* = 3) are launched for each conflict resolution variant in the hierarchy analyzed. No additional nested simulations are applied for the conflicts from the nested simulations: ${}^{1}C_2$, ${}^{1}C_3 \in Confl({}^{1}S_{1,1,1})$, ${}^{1}C_4 \in Confl({}^{1}S_{1,3,3})$, ${}^{1}C_6 \in Confl({}^{1}S_{5,1,3})$ and ${}^{1}C_8 \in Confl({}^{1}S_{7,2,3})$. Instead, they are addressed by applying some different approach (e.g., by using priority lists).

Figure 1(a) demonstrates the specific case of the hierarchy ${}^{0}H_0({}^{0}S_{0,0,r})$ consisting of a single replication of the main simulation (*maxLevel* = 0) with these conflicts: $^{\circ}C1$, $^{\circ}C2 \in Confl({}^{0}S_{0,0,r})$. RNS are not used to resolve such conflicts.

The algorithm discussed can be detailed in that the nested simulations can also resolve their conflicts recursively by using additional supporting nested simulations, as will be described later.

The problem on its own consists of identifying variants that should be tested for each conflict. The solution of this problem (implemented within the *VariantsGen* generator) cannot be

Table 1 Specifications of Symbols Related to the Control Algorithm 1

Symbols	Specifications
mainReplCount	Parameter specifying the number of main simulation replications (referred to simply as *main replications*).
nestReplCount	Parameter specifying the number of nested simulation replications (referred to simply as *nested replications*) executed for each resolution variant related to a relevant conflict.
stopCond	Parameter defining the terminating condition for the nested replication run; the condition may include either a specific moment of the simulation time (i.e., the lookahead duration) or the occurrence of a specific status/event.
maxVariants	Parameter determining the maximum permitted number of resolution variants tested for each conflict (but not the actual number of variants tested for a given specific conflict).
maxLevel	Parameter determining the maximum permitted number of nesting levels for a specific RNS with respect to the main simulation. If a conflict occurs in a nested replication, it can be resolved either by using nested simulations, too, or by using a different approach to the conflict resolution issue. In other words, the *maxLevel* parameter setting defines the maximum level to which the recursive simulation may be performed.
VariantsGen(i)	Software generator that will provide variants for the *i*-th conflict that will then be tested via nested simulations.
CrOptim(j)	Function evaluating the *j*-th conflict resolution variant based on a relevant optimization criterion.
${}^{d}S_{i,j,k}$	Symbol for a simulation experiment replication. For a nested simulation, this is the *k*-th replication for the *j*-th variant of resolution of the *i*-th conflict, where $k \in \langle 1 \ldots nestReplCount \rangle$, $j \in \langle 1 \ldots n \rangle$, $n \leq maxVariants$, $i \in N^{+}$ (the set of positive integers) and *d* corresponds to the *nestReplCount* parameter setting. If this is the *k*-th replication of the main ("non-nested") simulation, then $k \in \langle 1 \ldots mainReplCount \rangle$, $j = 0$, $i = 0$. The *d* (*depth*) value corresponds to the *maxLevel* parameter setting (i.e., replication ${}^{d}S_{i,j,k}$ belongs to an RNS-based simulating system applying a maximum of *d* levels of nesting).
${}^{d}H_b\ ({}^{d}S_{0,0,r})$	Symbol for a hierarchically ordered set of replications. The hierarchically highest (zeroth) level contains the *r*-th replication of the main simulation ${}^{d}S_{0,0,r}$ (element-root), the lower levels contain replications of the nested/recursive simulations (with respect to ${}^{d}S_{0,0,r}$). The *d* (*depth*) and *b* (*batch*) values correspond to the *maxLevel* and *nestReplCount* parameter settings. If the *r*-th replication of the main simulation ${}^{d}S_{0,0,r}$ does not use nested simulations, then the symbol for the set is ${}^{0}H_0\ ({}^{d}S_{0,0,r})$ and ${}^{0}H_0\ ({}^{d}S_{0,0,r}) = \{{}^{d}S_{0,0,r}\}$, $r \in \langle 1 \ldots mainReplCount \rangle$.
${}^{d}C_i\ (V_j)$	Symbol for the *i*-th conflict within a replication $X \in {}^{d}H_b({}^{d}S_{0,0,r})$ for a given *r*. After the conflict has been encountered, the calculation of replication *X* continues with its *j*-th solution variant (V_j).
$Confl({}^{d}S_{i,j,k})$	Symbol for a linearly ordered set of conflicts that occurred within replication ${}^{d}S_{i,j,k}$. The order reflects the timestamps of the conflicts.

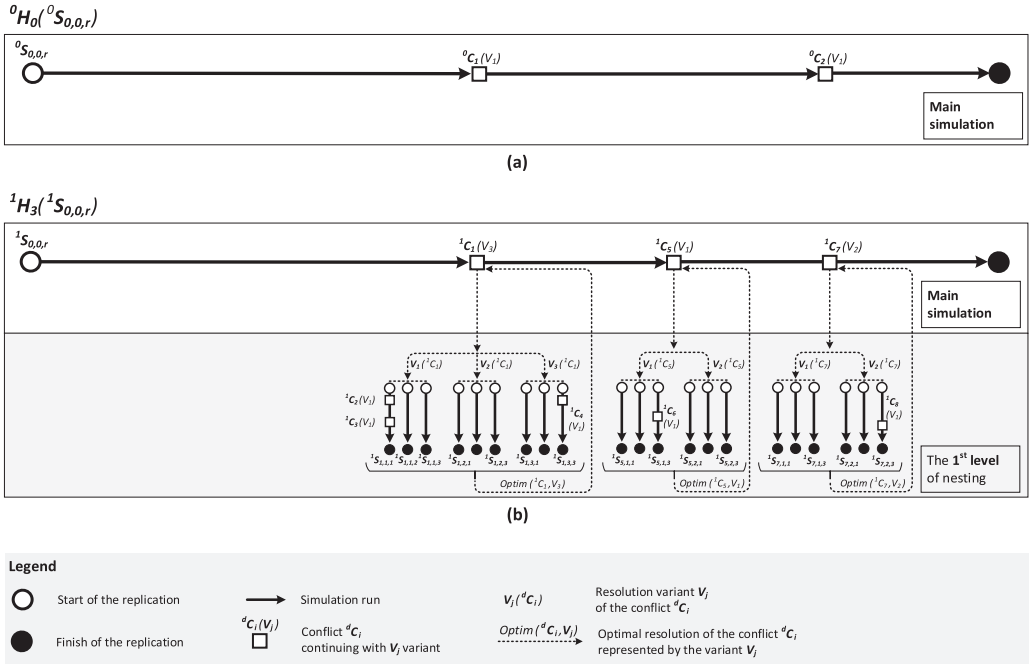

Fig. 1. Hierarchically ordered sets of completely computed replications: ${}^{0}H_0({}^{0}S_{0,0,r})$ and ${}^{1}H_3({}^{1}S_{0,0,r})$.

generalized appreciably because it always depends on the conflicting situation type and on the variant selection method, which in its outcome may be associated with solving a nontrivial problem.

Only such nested simulations are admissible in railway traffic simulators as are associated with technically and technologically permissible conflict resolution variants. Permissible variants take into account (i) the specific rail infrastructure topology tested, (ii) the interlocking system type used, and (iii) the traffic regulations covering the traffic examined.

3.2 Condition for Terminating Nested Simulations

The *stopCond* parameter generally defines the condition, the fulfillment of which will cause a nested replication to be terminated. This condition may be associated with the occurrence of a specific state in the simulation—a *termination state*—or may define a specific instant of the simulation time when the simulation is terminated. This instant is normally derived from a parameter specifying the *Lookahead Duration* and, hence, denoted *LD*. The lookahead duration is the simulation period available to the nested replication for executing its predictive computation.

Thus, if the *stopCond* parameter is related to the lookahead duration and, at the same time, the parameter $maxLevel > 1$, it must be specified how the lookahead duration is applied at lower levels of nesting. Generally, 3 basic strategies can be used:

- Constant lookahead durations
- Hierarchically reduced lookahead durations
- A combination of both

The first two strategies are described in detail next. They can be combined if appropriate for the specific problem being addressed. Sometimes, the combination of the lookahead duration and a

Table 2. Hierarchically Reduced LD for Selected Replications in Figure 5

Replication ID	Conflict ID	Conflict occurrence time	Lookahead duration	Applied to replications
${}^3S_{0,0,r}$	${}^3C_1$	09:00 a.m.	*Primary lookahead* $LD\,({}^3C_1) = 30$ minutes (1st level of nesting)	${}^3S_{1,1,1} \mid {}^3S_{1,1,2} \mid {}^3S_{1,1,3}$ ${}^3S_{1,2,1} \mid {}^3S_{1,2,2} \mid {}^3S_{1,2,3}$ ${}^3S_{1,3,1} \mid {}^3S_{1,3,2} \mid {}^3S_{1,3,3}$
${}^3S_{1,1,1}$	${}^3C_2$	09:05 a.m.	*Secondary lookahead* $LD\,({}^3C_2) = 25$ minutes (2nd level of nesting)	${}^3S_{2,1,1} \mid {}^3S_{2,1,2} \mid {}^3S_{2,1,3}$ ${}^3S_{2,2,1} \mid {}^3S_{2,2,2} \mid {}^3S_{2,2,3}$
${}^3S_{2,2,3}$	${}^3C_3$	09:20 a.m.	*Tertiary lookahead* $LD\,({}^3C_3) = 10$ minutes (3rd level of nesting)	${}^3S_{3,1,1} \mid {}^3S_{3,1,2} \mid {}^3S_{3,1,3}$ ${}^3S_{3,2,1} \mid {}^3S_{3,2,2} \mid {}^3S_{3,2,3}$ ${}^3S_{3,3,1} \mid {}^3S_{3,3,2} \mid {}^3S_{3,3,3}$ ${}^3S_{3,4,1} \mid {}^3S_{3,4,2} \mid {}^3S_{3,4,3}$
${}^3S_{3,3,1}$	${}^3C_4$	09:22 a.m.	*No lookahead*	–
${}^3S_{3,3,1}$	${}^3C_5$	09:28 a.m.	*No lookahead*	–
${}^3S_{1,1,1}$	${}^3C_6$	09:25 a.m.	*Secondary lookahead* $LD\,({}^3C_6) = 5$ minutes (2nd level of nesting)	${}^3S_{6,1,1} \mid {}^3S_{6,1,2} \mid {}^3S_{6,1,3}$ ${}^3S_{6,2,1} \mid {}^3S_{6,2,2} \mid {}^3S_{6,2,3}$ ${}^3S_{6,3,1} \mid {}^3S_{6,3,2} \mid {}^3S_{6,3,3}$

Legend
$LD\,({}^dC_i)$ – *Lookahead durations* for replications inspecting resolutions related to ${}^dC_i$

specific termination state may also be feasible, for instance, by applying one strategy at the first nesting level and the other strategy at the lower nesting levels.

Constant lookahead durations

The *constant lookahead duration* strategy uses a simple approach in which a single, a priori–defined lookahead duration is assigned to all replications of nested simulations, irrespective of the level of nesting or any other parameter. As a consequence, the predictive simulations at lower levels of nesting examine a more remote future than the hierarchically superior simulations.

Hierarchically Reduced Lookahead Durations

This strategy uses the following procedure. If a conflict that should also be addressed by using RNS occurs during the computation of a nested replication, the lookahead duration for a simulation at a lower hierarchy level is reduced by the fraction that has already been "spent" at the previous hierarchy levels. Thus, this approach uses *hierarchically reduced lookahead durations*, and its application can be demonstrated on the example of replications selected from the set ${}^3H_3({}^3S_{0,0,r})$ shown in Figure 5. The lookahead duration that is set for replications at the first level of nesting is referred to as the *primary lookahead duration*. A conflict ${}^3C_1$ occurred within replication ${}^3S_{0,0,r}$ of the main simulation (at the hierarchy level 0) at the 09:00 hours of the simulation time. This triggers nested simulations at the first level of nesting (verifying the solution variants V_1, V_2, and V_3), for which the primary lookahead duration has been set at 30 minutes. The assigned lookahead durations for selected replications of nested simulations related to the primary investigation of the conflict (${}^3C_1$) resolution are listed in Table 2. The table shows that the *secondary lookahead durations* for the replications at the second level of nesting are 25 minutes when resolving the conflict ${}^3C_2$ and 5 minutes when resolving the conflict ${}^3C_6$. The *tertiary lookahead duration* at the third level of nesting (when resolving the conflict ${}^3C_3$) is 10 minutes. All replications of the nested simulations listed in Table 2 run till 09:30 hours of the simulation time. Nested simulations are

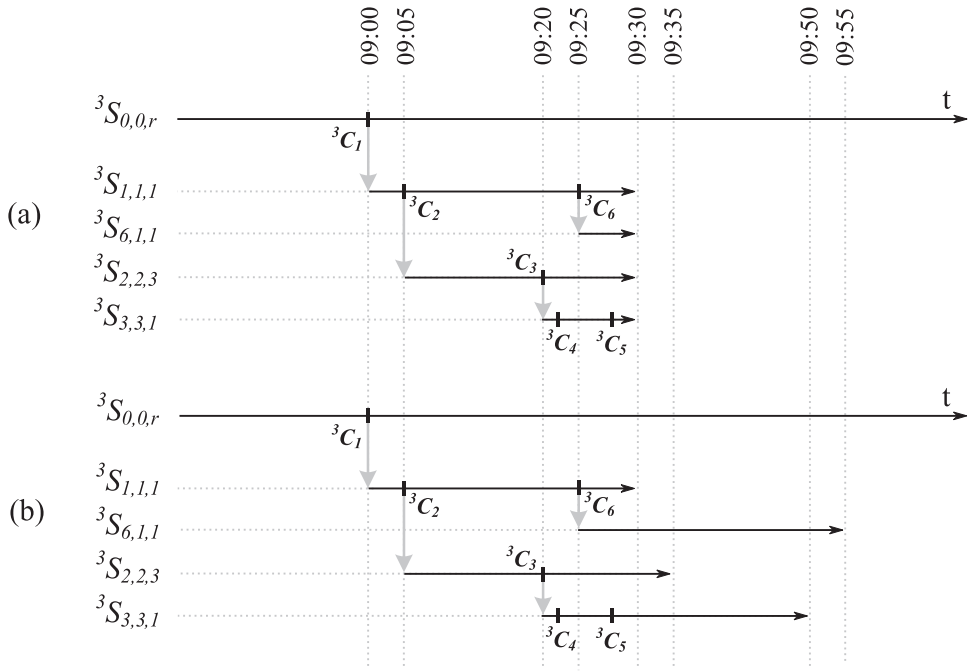

Fig. 2. Illustration of (a) hierarchically reduced lookaheads and (b) constant lookaheads.

not used to resolve conflicts encountered in replications at the lowest hierarchy levels (e.g., $^3C_4$ or $^3C_5$). An alternative decision-making support method is used instead.

For a graphic comparison of the consequences of these strategies, selected replications (based on Table 2) representing the 1st, 2nd and 3rd levels of nesting are shown in Figure 2, whose top part (a) illustrates hierarchically reduced lookahead durations while the bottom part (b) illustrates constant lookahead durations.

3.3 Parallel Computations Related to Sequential Simulators Applying RNS

For nested simulations, the separate and extensive problem is associated with the method of their computation, which is a complex computing job. The complexity of this problem is primarily affected by the following factors:

- *Number of conflicts* (within both the main simulation and the nested simulations), which are resolved through testing of the variants by using predictive nested simulations.
- *Number of variants* tested during the conflict resolution job.
- Number of replications of the nested simulations made for each variant tested.
- *Lookahead duration* (or the terminating condition) applied in the nested simulations.
- Number of main simulation replications.

The occurrence of conflicts within the nested simulations can be associated with recursive execution of additional nested simulations, which brings about exponential growth of the number of launched simulation runs. The recursive propagation of the simulation can be limited through the above-mentioned *maxLevel* parameter. Conflict resolution in nested simulations working on the lowest permitted nesting level is based on the application of a "non-simulation" deterministic approach (e.g., by using priority lists, artificial neural networks, multicriterial decision-making methods, etc.).

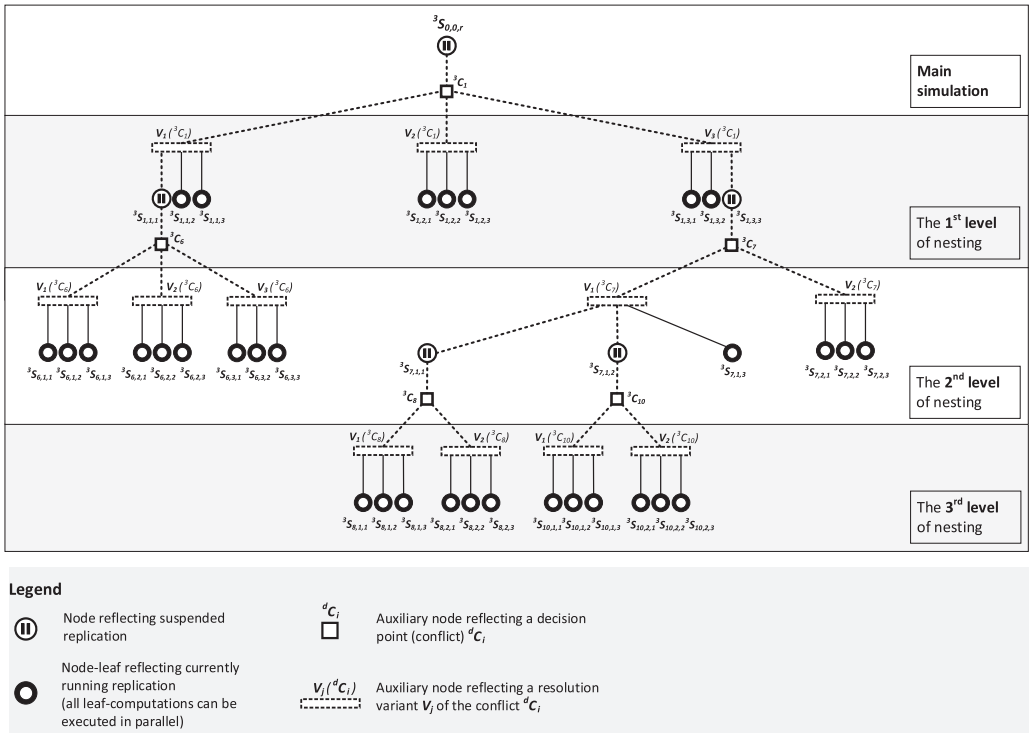

Fig. 3. Dynamic tree reflecting the current state of the computation regarding ${}^3H_3({}^3S_{0,0,r})$ shown in Figure 5.

Technical implementation of a nested simulation can benefit from the fact that the nested replications (working on the same nesting level) run autonomously without affecting one another. Hence, it is technically feasible, through appropriate implementation, to support *multisimulations*, that is, parallel processing of independent simulations/replications. Appropriate hardware means are supposed to be utilized for this.

The evolution of one main replication (${}^dS_{0,0,r}$) within which RNS are applied in support of decision-making can be presented by using the *dynamic rooted tree* $Y^t({}^dS_{0,0,r})$. For any specific real-time instant t ($t \in \langle t_s, t_f \rangle$), where t_s is the time replication, ${}^dS_{0,0,r}$ is started and t_f is the time in which this replication is completed, it holds that the tree contains (i) a *node-root* mirroring replication ${}^dS_{0,0,r}$ and (ii) potentially additional nodes corresponding to all started and not yet completely computed replications of nested simulations from the set ${}^dH_b({}^dS_{0,0,r})$, where the parameters d and b are fixed.

When starting computation of new nested replications (at a real-time instant t, $t \in \langle t_s, t_f \rangle$), appropriate new leaves must be put into the tree $Y^t({}^dS_{0,0,r})$. The inserted leaf is a descendant of the parent node that initiated the new replication. Once the computation of each replication—corresponding to the *node-leaf* in the tree $Y^t({}^dS_{0,0,r})$ at the real time t ($t \in \langle t_s, t_f \rangle$)—is completed, the leaf is removed from the tree. All replications corresponding to *nodes-leaves* of the tree $Y^t({}^dS_{0,0,r})$, $t \in \langle t_s, t_f \rangle$, are mutually independent and potentially can be computed in parallel.

By way of illustration, one of the possible configurations of the tree $Y^\tau({}^3S_{0,0,r})$ is shown in Figure 3. This tree mirrors (for a real-time instant τ) the possible development of the computation of the set of replications ${}^3H_3({}^3S_{0,0,r})$, illustrated graphically in Figure 5. All of the 32 replications pertaining to the leaves of $Y^\tau({}^3S_{0,0,r})$ are computed in parallel (the computation of 5 replications is currently suspended because the conflicts are being resolved by using RNS).

The evolution of a complete simulation experiment, which entails a given number of main replications ($^{d}S_{0,0,r}$, $r = 1 \ldots mainReplCount$), can be represented by means of a *dynamic forest*. This forest always includes *dynamic rooted trees* reflecting the started and not yet completed main replications. All main replications are independent and potentially can be computed in parallel.

Multisimulations within a computer node can be performed by using more than one process or more than one thread, whereby the various physical or virtual *CPU* cores can be used for more efficient parallel processing of the simulations. Another option for performing simulation consists of distributing the computations to several computer nodes, typically interconnected by a computer network or through the Internet. As a prerequisite for multisimulations, the simulator must support:

- Saving the current status of the replication
- Modification of the saved status for preparations for launching the appropriate conflict resolution variant
- Serialization of the saved status to a data file

When saving (cloning) the status of a sequential simulation, the simulation process itself must be suspended in order to ensure a consistent status of the simulation during the cloning procedure. A complete instantaneous copy of the entire simulation state space is then created. The cloning can be accomplished in the simulator by using various techniques, for example, by using a component providing memory management (related to a simulator's state space) or by mapping the state variables using metadata.

The simulator MesoRail [35] (used in the case study described in Section 5) saves the status in the variables (attributes) of the simulation objects (agents). Each state variable includes annotation (metadata) determining the state variable identifier, method of its serialization/deserialization, and additional auxiliary information. Each object is also capable of creating a clone, which implicitly shares the unchanging state variables (whose values do not change during the simulation process).

The data files created must be suitably transferred to the various processes/computation nodes. For instance, the following can be used depending on the computation method applied:

- Shared memory for computations within a computation node
- Shared file repository or communication via network sockets for distributed computations

The need for communication between or synchronization of the nested simulations does not arise during the computation itself. When a replication of a nested simulation is over, information on the results must be transferred to the higher-level replication. To sum up, the following holds for any sequential simulator applying RNS:

- Any individual replication (of a main simulation or a nested simulation) is executed through sequential computation.
- If a conflict is encountered during a replication and is addressed via RNS, the replication is suspended and additional nested replications at a lower hierarchy level are initiated (by using cloning) and potentially started immediately.
- All main simulation replications can be computed in parallel.
- Mutually independent replications of nested simulations (under the same main replication $^{d}S_{0,0,r}$) can also be computed in parallel. Such replications can be identified—at a given real-time instant t—as leaves of the *dynamic rooted tree* $Y^{t}(^{d}S_{0,0,r})$, which is a model of the computation development.

Algorithm 1 can be additionally detailed with respect to the parallelization of the replication computations. The pseudocode of Algorithm 2 represents the control program for the parallel

Table 3. Specifications of Symbols Related to the Control Algorithm 2

Symbols	Specifications
MainSimulation()	Main program launching the parallel computation of the main simulation replications.
RunRepl(i,j,k,L)	Recursive subprogram launching replication ${}^{d}S_{i,j,k}$ working on the *L*-th level of nesting.
Initialize(i,j,k,L)	Subprogram for initialization of replication ${}^{d}S_{i,j,k}$ working on the *L*-th level of nesting.
SaveResult(i,j,k,L)	Subprogram for saving the results of replication ${}^{d}S_{i,j,k}$ working on the *L*-th level of nesting. The results include replication evaluation according to the relevant optimization criterion.
MultirunVariant(i,j,L)	Subprogram for the parallel computation of replications ${}^{d}S_{i,j,k}$ ($k = 1 \ldots nestReplCount$) belonging to the *j*-th variant of resolution of the *i-th* conflict.
Simulate()	Basic routine of the simulation engine implementing the life cycle of the replication run.
SelectOptim(i)	Function selecting the optimum variant from among the simulated variants for resolution of the *i*-th conflict (each variant is evaluated by using the function *CrOptim*).
EstimOptim(i)	Function estimating (by the "non-simulating" deterministic approach) the optimum resolution of the *i*-th conflict.
FinishPoint(), ConflictPoint()	Boolean functions deciding whether a conflict occurred in the running replication/whether the condition for termination of the replication is met (based on the parameter *stopCond*).
ContinueVariant(i,j)	Subprogram setting parameters for continuation of the run of a replication (following occurrence of the *i*-th conflict) for the conditions of the *j*-th resolution variant.
conflictID	Global variable (conflict counter).
currConflict, bestVariant, k, r, v	Auxiliary variables.

computations related to sequential simulators applying RNS. It uses some of the symbols listed in Table 1 and introduces additional symbols as defined in Table 3.

The keywords "parbegin" and "parend" are used in the pseudocode. Computations of the subprograms whose calls are encapsulated through those keywords are made in parallel (if the appropriate computation means are available).

Examples of completed runs of Algorithm 2 with different settings of the parameter *maxLevel* are graphically outlined in Figures 4 and 5. The symbols are as in Figure 1.

3.4 Possibilities of Sharing Computations

From the viewpoint of *one arbitrary set of replications* ${}^{d}H_b({}^{d}S_{0,0,r})$, the RNS method (applied to the resolution of conflicts) effectively uses/shares computations among selected replications. When a replication $X \in {}^{d}H_b({}^{d}S_{0,0,r})$ is suspended at an instant t of the simulation time because of the occurrence of a conflict (decision point), replication X is cloned for triggering the predictive nested simulations. Those nested simulations always implicitly share the results of existing computations

ALGORITHM 2: Parallel executions of replications

```
Parallel executions of replications related to sequential simulators applying RNS
01  program MainSimulation()
02      conflictID ← 1
03      for r = 1 to mainReplCount do
04          parbegin
05              Initialize(0,0,r,0)
06              RunRepl(0,0,r,0)
07          parend
08      end
09  end
10
11  function RunRepl(i,j,k,L)
12      while not FinishPoint() do
13          repeat
14              Simulate()
15          until FinishPoint() or ConflictPoint()
16          if FinishPoint() then
17              SaveResult(i,j,k,L)
18              exit
19          else
20              currConflict ← conflictID
21              conflictID ← conflictID + 1
22              if L < maxLevel then
23                  for v = 1 to VariantsGen(currConflict) do
24                      parbegin
25                          MultirunVariant(currConflict,v,L+1)
26                      parend
27                  end
28                  bestVariant ← SelectOptim(currConflict)
29              else
30                  bestVariant ← EstimOptim(currConflict)
31              end
32              ContinueVariant(i,bestVariant)
33          end
34      end
35  end
36
37  function MultirunVariant(i,j,L)
38      for k = 1 to nestReplCount do
39          parbegin
40              Initialize(i,j,k,L)
41              RunRepl(i,j,k,L)
42          parend
43      end
44  end
```

of replications that are hierarchically superior to replication X. This principle can be illustrated on the example of a currently running computation (graphically shown in Figure 3) as follows: replications ${}^3S_{8,1,1}$, ${}^3S_{8,1,2}$, ${}^3S_{8,1,3}$, ${}^3S_{8,2,1}$, ${}^3S_{8,2,2}$, and ${}^3S_{8,2,3}$ share a portion of the computation of the replication ${}^3S_{7,1,1}$. The last-mentioned replication follows up the computation of replication ${}^3S_{1,3,3}$, which uses the existing results of the replication ${}^3S_{0,0,r}$.

Other options to improve the effectiveness of computations of replications from *multiple sets* ${}^dH_b({}^dS_{0,0,r})$, $r = 1 \ldots mainReplCount$ (where the parameters d and b are fixed) are rather limited both as regards sharing of parts of their computations and as regards elimination (pruning) of some non-promising nested simulations. This is so because all replications of both the main

Table 4. Sharing Computations by Replications from ${}^{1}H_3({}^{1}S_{0,0,r})$ and ${}^{2}H_3({}^{2}S_{0,0,r})$

Basic scenario *Sc0203*	Depth-extended scenario *Sc0303*	
Replications from the set ${}^{1}H_3({}^{1}S_{0,0,r})$	Matching replications from the set ${}^{2}H_3({}^{2}S_{0,0,r})$	Common computation
${}^{1}S_{0,0,r}$	${}^{2}S_{0,0,r}$	until conflict ${}^{1}C_1$ (${}^{1}C_1 = {}^{2}C_1$)
${}^{1}S_{1,1,2} \mid {}^{1}S_{1,1,3}$ ${}^{1}S_{1,2,1} \mid {}^{1}S_{1,2,2} \mid {}^{1}S_{1,2,3}$ ${}^{1}S_{1,3,1} \mid {}^{1}S_{1,3,2}$	${}^{2}S_{1,1,2} \mid {}^{2}S_{1,1,3}$ ${}^{2}S_{1,2,1} \mid {}^{2}S_{1,2,2} \mid {}^{2}S_{1,2,3}$ ${}^{2}S_{1,3,1} \mid {}^{2}S_{1,3,2}$	complete replications
${}^{1}S_{1,1,1}$	${}^{2}S_{1,1,1}$	until conflict ${}^{1}C_2$ (${}^{1}C_2 = {}^{2}C_2$)
${}^{1}S_{1,3,3}$	${}^{2}S_{1,3,3}$	until conflict ${}^{1}C_4$ (${}^{1}C_4 = {}^{2}C_4$)

simulation and the nested simulations have differently set the seeds of their pseudorandom number generators responsible for producing the occurrences of random phenomena during the simulations. This means that the evolutions of all replications are principally different and the results obtained from all of them must be taken into account in the computation of the statistical traffic indicators reflecting the appropriate optimization criterion.

The possibility of sharing parts of computations can potentially be used as well for selected replications computed within different scenarios of a simulation experiment. This concept can be applied with success, for example, to what is called *depth-extended scenarios*. If a scenario *ScA* is looked upon as a *basic scenario*, then a scenario *ScB* is a depth-extended scenario to *ScA* if:

- the scenarios differ in the value of the *maxLevel* parameter, specifically $maxLevel_{ScB} > maxLevel_{ScA}$;
- the same procedure is applied to the two scenarios when setting different pseudorandom number generator seeds before starting to use them within newly initiated replications (of both the main simulation and the nested simulations), in other words, two matching replications in the different scenarios have identically set the seeds;
- all the remaining parameter values are identical in the two scenarios.

Computation sharing among depth-extended scenarios can be illustrated on the demonstration sets ${}^{1}H_3({}^{1}S_{0,0,r})$, ${}^{2}H_3({}^{2}S_{0,0,r})$, and ${}^{3}H_3({}^{3}S_{0,0,r})$, which are related to the *r*-th replications of main simulations belonging to the scenarios *Sc0103, Sc0203*, and *Sc0303* ($maxLevel_{Sc0103} = 1$, $maxLevel_{Sc0203} = 2$ and $maxLevel_{Sc0303} = 3$). The sets are depicted graphically in Figures 1(b), 4, and 5, respectively. If a complete computation is first made according to the basic scenario *Sc0103* that is associated with the computation of replications from the set ${}^{1}H_3({}^{1}S_{0,0,r})$, this computation can be extended. When the depth-extended scenarios *Sc0203* and *Sc0303* associated with the computation of replications from the sets ${}^{2}H_3({}^{2}S_{0,0,r})$ and ${}^{3}H_3({}^{3}S_{0,0,r})$ are being computed, selected computations from the basic scenarios potentially can be shared. This selected computation sharing can be demonstrated in the example (see Table 4) in which replications from ${}^{2}H_3({}^{2}S_{0,0,r})$ use parts of computations of replications from ${}^{1}H_3({}^{1}S_{0,0,r})$.

The testing exercise for the different methods of implementation of nested simulations included testing of computations related to the *depth-extended scenarios* (with the aim to optimize selected computations). The computation experiments were aimed to test selected scenarios of the case study described in Section 5. Data obtained from simulation testing experiments [36] revealed that this procedure does not improve appreciably the effectiveness of the computations based on the use of RNS, and it was not applied within the case study described later. The main reasons why the above procedure was not recommended are as follows:

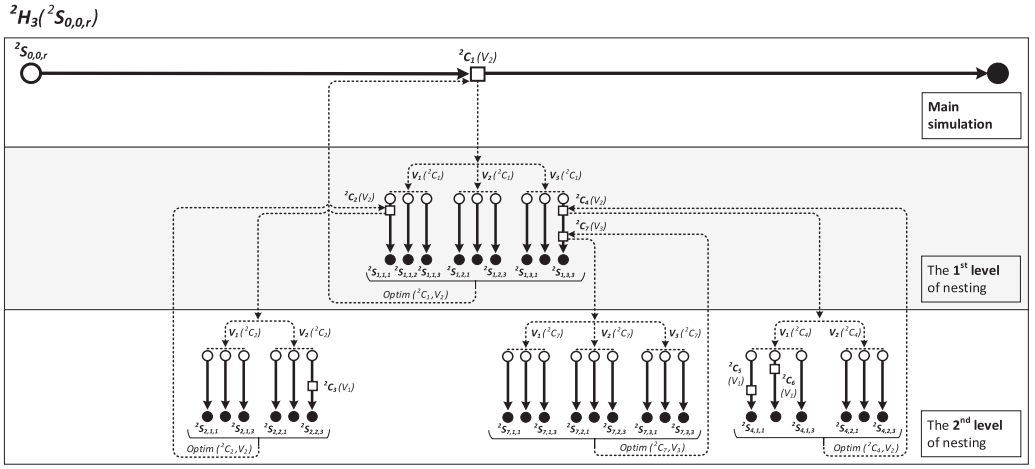

Fig. 4. Hierarchically ordered set of completely computed replications: ${}^{2}H_3({}^{2}S_{0,0,r})$.

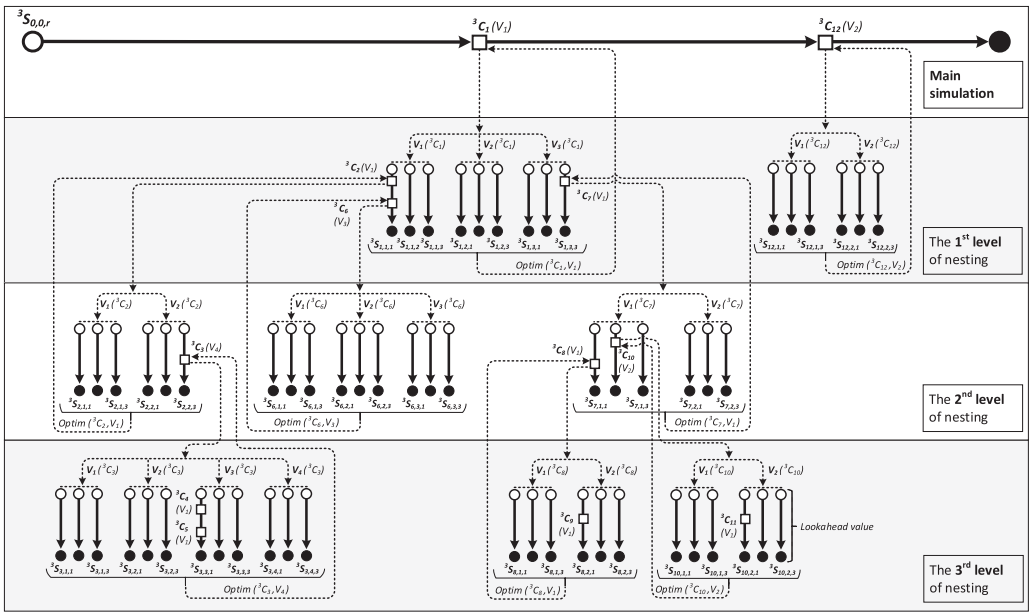

Fig. 5. Hierarchically ordered set of completely computed replications: ${}^{3}H_3({}^{3}S_{0,0,r})$.

- High demands put on the memory for saving (although temporarily) a rather high number of simulator states
- Rather high time demands due to the frequent loading of the simulator states.
- Relatively low computational time savings against the computation within which the above procedure is not applied

Nevertheless, the applicability of the concept of *depth-extended scenarios* can always be re-examined during the modelling of other systems in order to determine whether marked computation time savings can be achieved.

4 GUIDELINES FOR EXPLOITING NESTED SIMULATIONS WITHIN APPLICATION PROJECTS

The main goal of the recommended guidelines described later is to apply the best possible configuration of automated support to optimization/decision-making by using RNS for a defined time limit that is available for processing a simulation study/project. The main task is to select good solutions/decisions when conflict situations are encountered. The solutions to resolve conflict situations (by using RNS) should match those that would (optimally) be adopted by experienced dispatchers.

Efficient use of the computing resources available for the simulations using the RNS approach is a next goal. Once the first phase of the recommended procedure is over, the computer resources (computer cluster, a high-performance PC, etc.) are divided into *logical processing units—LPUs*—each LPU being assigned for complete computation of one replication of the main simulation. This principle of *static assignment of the computing resources* is aimed to support the execution of parallel computations of multiple main replications. This approach can be applied on a single computer, on a computer cluster, or in a cloud environment. Moreover, no specific properties of computing resources or the presence of specific software or a particular tool for planning processes are generally required.

Work on a simulation project/study is always associated with the analysis of the output indicators emerging from the main replications within the set M ($M = \{ {}^{d}S_{0,0,r} \mid r = 1 \ldots mainReplCount \}$). Before executing a simulation using RNS, a number of input parameters (i.e., the *RNS configuration*) must be set. Some of the parameters (*stopCond, mainReplCount*) must be set for the complete simulation study; other parameters are derived based on the time available for performing the whole simulation study (*totalTimeLimit*) and on the hardware available for the computations of the replications.

The following proposed procedure is divided into 3 phases (illustrated in Figure 6): the initial, *probing phase* includes rapid (rough) assessment of the computational complexity (and hardware requirements for the simulations) for various RNS configurations. Based on the result of this phase, the computer tools are divided into LPUs. This division into LPUs is applied within the 2 subsequent phases for parallel computations of multiple main replications. In the second, *pilot phase*, pilot computations are executed for selected simulation experiment *scenarios* analyzing the quality of the resulting solution and the demands put by the scenarios on the computing resources. The *final phase* includes completion of computations belonging to the selected ("winner") scenarios from the pilot phase in order to obtain statistically significant results (characterizing the operation examined).

The procedure is used to examine one specific version of the system investigated. Where the simulation study tests multiple system versions, the procedure must be performed repeatedly. From the practical aspect, a time limit must be set for the examination/optimization of one system version (*versionTimeLimit*). The total study/project time limit (*totalTimeLimit*) can be uniformly divided into stages, each devoted to the examination of one system version (within the project) unless reasons exist why the versions can be expected to vary in time requirements. The phases of the proposed procedure are described in the following sections.

4.1 Probing Phase

The preparatory *probing phase* includes a small number of differently parameterized simulation experiments (with a small number of main replications) performed to obtain a basic presumption of the computational and disk space demands of future complex computations. The experiments use an appropriate lookahead time (or multiple lookahead times) for the RNS, taking into account

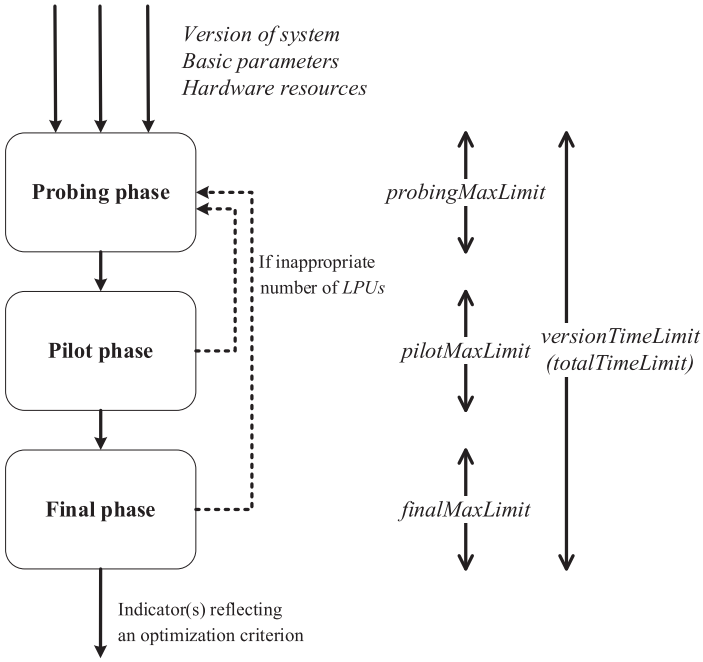

Fig. 6. Procedure of RNS-based investigation reflecting a particular version of the system examined.

the nature of the system being examined. Maximum limits can be set in this phase both for the computation time of one main replication (*replMaxLimit*) and for the total duration of this phase (*probingMaxLimit*).

The depth-extended scenario presented in Section 3.4 can be potentially employed during selected experiments, thereby achieving computing time savings.

The probing process forms the basis for a draft concept of the technical solution of parallel calculations on the computer system available. This draft concept reflects the requirements for (i) the highest possible parallelization of the computations within the simulation experiment and (ii) an adequate memory space needed for the computations. In this way, the available computing resources (computer cluster, multicore PC, etc.) are divided into LPUs. Each LPU has available system tools adequate for simulation of one main replication. It is assumed that each *LPU* addresses parallelization of the computations of nested replications (e.g., by using multithreading) under the umbrella of a given main replication.

Where feasible, this technical solution is then applied within the entire experimental stage (i.e., during the simulation testing of all of the system versions examined).

4.2 Pilot Phase

The *pilot phase* focuses on the execution of a fraction of the main replications for all scenarios planned for executions. The scenarios (RNS configurations) differ specifically in (i) the numbers of nesting levels (set through the *maxLevel* parameter) and (ii) the numbers of replications executed within nested simulations (set through the *nestReplCount* parameter).

The executions of main replications (the tasks) are planned by using the *FCFS* (*First Come First Served*) approach always to the first free LPU.

In order to specify selected scenarios, the computation of which should be completed within the final phase, it is necessary to evaluate the appropriate traffic indicator values (criterial function values). In addition, the expected computational (time) demands associated with the completion of the computation of the required number of main replications (*mainReplCount*) for each investigated scenario must be estimated. A coarse estimate of the total duration (${}^{estim}T_N$) of the future final experiment computation (consisting of N main replications and applying parallel computations on m LPUs) is calculated by extrapolation based on the durations of the various main replications of the appropriate pilot experiment. The pilot experiments use n replications of the main simulations ($N \gg$ n). For each experiment, the durations of the computations of its main replications are obtained. They are used to calculate the mean duration of one main replication (${}^{mean}t_n$), including the 95% confidence interval. The duration estimate for one replication is calculated as ${}^{repl}t = {}^{upper}t_n + {}^{cost}t$, where ${}^{upper}t_n$ is the upper endpoint of the confidence interval (regarding ${}^{mean}t_n$) and ${}^{cost}t$ represents overhead time needed by the planner for (i) starting up a new process, or new replication; (ii) harvesting the process results; (iii) terminating the process; and (iv) the planning process itself. The calculation of the (pessimistic) estimate can be formalized as:

$$ {}^{estim}T_N = \frac{N \times {}^{repl}t}{m} + {}^{repl}t \tag{1} $$

Maximum time limits can be set in this phase both for the computation time of one main replication (*pilotReplMaxLimit*) and for the total duration of this phase (*pilotMaxLimit*).

4.3 Final Phase

First, the *final phase* serves to evaluate the estimates from the preceding phase and to select suitable simulation scenarios for completing their computations. Then, selected scenarios are completely computed and their outcomes are used for traffic evaluation of the tested version of the system examined.

If the initially proposed technical solution of parallel computations does not work well for some versions of the system (e.g., because of inadequate memory capacity), the probing experiments must be repeated. Based on the results, a different approach to the parallelization must be selected.

5 CASE STUDY

This case study is built on the contractual research project entitled "A New Methodology of Railway Station Capacity Assessment," in which the authors of the present article were engaged. The project was developed by the Faculty of Electrical Engineering and Informatics, University of Pardubice, for the Czech governmental agency SŽDC (Rail Infrastructure Administration) between 2014 and 2016.

The case study analyzed railway traffic through a minor passenger station (Zdice, Czech Republic), for a specific timetable. The simulation experiments used an optimization approach based on minimization of a certain operational railway traffic indicator (the total train delay increment) with regard to the defined system.

The optimization is achieved through computer simulation incorporating decision-making support regarding conflict resolution by using reflective nested simulations [34]. The conflict type examined was related to the requests for allocation of a specific station track for a given time window coming from more than one (potentially delayed) train.

The simulating system utilized within the case study is composed of three subsystems.

(a) The infrastructure subsystem (the layout of which is shown in Figure 7) was formed within the frame of an associated editor (called *TrackEd*). That subsystem realistically mirrors the

Table 5. Selected Characteristics of the Trains

Train type	Engines count/ wagons count	Train length	Train line	Intervals between trains	Probab.of delay occurrence (Bernoulli)	Mean delay (Exponential)
Express	1/7	202 m	*East* → *Central* → *West*	30 min	0.50	420 s
Express	1/7	202 m	*West* → *Central* → *East*	30 min	0.50	420 s
Regio	2/4	158 m	*East* → *Central* → *West*	10 min	0.33	270 s
Regio	2/4	158 m	*West* → *Central* → *East*	10 min	0.33	270 s
Regio	1/2	79 m	*East* → *Central* → *South*	30 min	0.33	270 s
Regio	1/2	79 m	*South* → *Central* → *East*	30 min	0.33	270 s
Cargo	1/22	336 m	*East* → *West*	60 min	0.50	1800 s
Cargo	1/22	336 m	*West* → *East*	60 min	0.50	1800 s

slope and curve parameters of each track in the appropriate data layer and includes the following structures:

- The track layout of the station, referred to in the case study as Central. Two platforms in particular are used for passenger trains: Platform 2 (with the adjacent tracks #1 and #3) and Platform 3 (with the adjacent tracks #2 and #4). Either platform is 300 m long.
- Rail lines to the neighboring boundary stations referred to as South, West, and East. The line to the South station is a single-track line; the lines to the West and East stations are two-track lines. The total distance between the West and East boundary stations is 20 km.
- Simplified track layouts of the *South, West*, and *East* boundary stations.

(b) Passenger trains predominate in the traffic subsystem, although freight trains also contribute. The timetable (set up by an SŽDC railway expert) is also defined, characterizing traffic during the rush hours of 08:00 to 10:00 a.m. (Figure 8) and encompassing 46 trains in total. Matching the traffic timetable, a static station track occupation plan is available for the *Central* station, primarily stating which station tracks should be allocated to specific trains (see Figure 8). Ideally, tracks #1, #2, and #3 should be allocated to the trains. Six station tracks are available for alternative allocations in case the plan cannot be adhered to in some traffic situations. Tracks #6 and #8 are primarily intended for alternative allocation to freight trains while the platform tracks #1, #2, #3, and #4 are intended for variant allocation to passenger trains. The trains entering the system are burdened by random delays, the characteristics of which were specified by SŽDC experts (note that the vast majority of the experiments used stochastic simulations). Sets of acceptable alternative train routes (passing via different station tracks) within the *Central* station infrastructure were defined for each train to enable trains to be routed through different trajectories in different traffic situations. Two types of trains for passenger transport were distinguished: passenger trains (for regional transport) and express trains (for long-distance transport). The basic train characteristics are listed in Table 5.

(c) Focus in the control procedure subsystem is mainly on decision-making support in the resolution of conflicts relating to the allocation of station tracks to the trains entering the station. Two approaches are available for this:

- *Nested simulations*, primarily testing all the currently acceptable variants of station track allocation to an arriving train. In addition, simulated is also the variant in which the arriving train will "only" wait until the track initially determined for it in the static

Table 6. Priority Lists with Alternatives of Station Track Allocation to Trains

Train line	Train type	Count of trains	Train ID	Train line route		Static priority lists (station tracks)			
				From the station (line track)	To the station (line track)	1.	2.	3.	4.
R1	*Express*	5	750, 752, …, 758	*East* (*E2*)	*West* (*W2*)	#2	#4	#1	#3
R2	*Express*	5	751, 753, …, 759	*West* (*W1*)	*East* (*E1*)	#1	#3	#2	#4
Os1	*Regio*	12	7800, 7802, …, 7822	*East* (*E2*)	*West* (*W2*)	#2	#4	#1	#3
Os2	*Regio*	12	7801, 7803, …, 7821	*West* (*W1*)	*East* (*E1*)	#1	#3	#2	#4
Os3	*Regio*	4	7900, 7902, …, 7906	*East* (*E2*)	*South* (*S1*)	#2	#4	#1	#3
Os4	*Regio*	4	7901, 7903, …, 7907	*South* (*S1*)	*East* (*E1*)	#3	#1	#2	#4
Pn1	*Cargo*	2	60000, 60002	*East* (*E2*)	*West* (*W2*)	#2	#4	#6	#8
Pn2	*Cargo*	2	60001, 60003	*West* (*W1*)	*East* (*E1*)	#1	#2	#6	#8

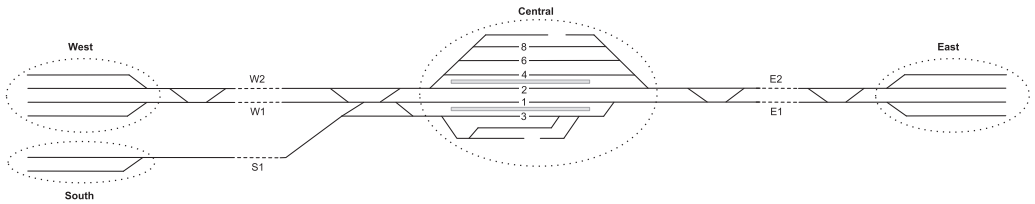

Fig. 7. Track layout overview reflecting an area of the simulated system.

track occupation plan is made free. The variant that provides the best result (obtained by using the *CrOptim* function) is selected for resolving the conflict.

- *Static priority lists*, containing all of the generally acceptable variants of station track allocation to specific arriving trains. The lists are sorted by element/variant priorities, the priority of a variant being the higher it is, the more convenient it is from the traffic aspect. The highest-priority variant matches the solution that complies with the *station track occupation plan*. The lists for the case study were prepared in line with the expert recommendation. The feasible variant that has the highest priority was selected from the list for each arriving train.

Nested simulations were primarily used for resolving these conflicts, whereas priority lists were used to select the solutions for conflicts occurring only in the last permitted level of nesting (defined by the *maxLevel* parameter). A sample of priority lists set up for station track allocation in the *Central* station is shown in Table 6. The track labelling is shown in Figure 7. Note that the priority lists are used in two ways in this case study:

- Selection of *one* most suitable ("non-simulation") currently feasible conflict resolution is applied in the replications on the lowest hierarchy level. In such situations, the priority lists are the only decision-making supporting tools.
- In the remaining replications, the selection of *all* currently feasible solutions is performed above the priority lists. The solutions so selected are then tested within replications of nested simulations. The selection is the task of the software variant generator *VariantsGen*, mentioned in Table 1. Nested simulations are applied in support of decision-making.

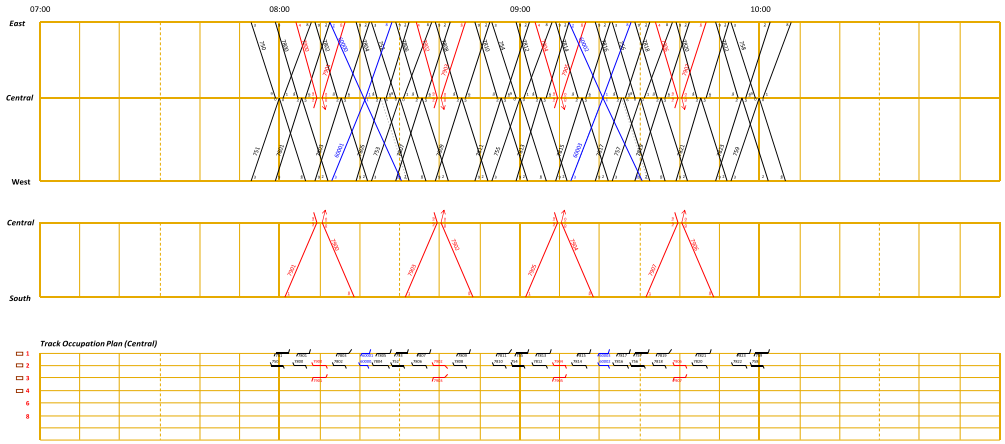

Fig. 8. Graphic timetable and track occupation plan for morning rush hour.

5.1 Basic Information on the Simulation Experiments

As regards the completed simulation experiments aimed at testing the railway traffic characteristics in the *Central* passenger station (applying the defined timetable), let us sum up the basic information on the simulating system built within the MesoRail tool.

- The simulation experiments mirror the traffic during the morning rush hour (08:00-10:00 a.m.) in the *Central* passenger station. In the timetable used (Figure 8), the simulating system is entered first by trains in the *West* and *East* boundary stations (at 07:53 a.m.) and is left as the last by the train in *East* boundary station (at 10:08 a.m.).
- A *deterministic simulation experiment* was performed for initial testing of the ideal application of the timetable, with the traffic free from any train delay burden. The experiment gave evidence that the timetable was conflict free.
- In *stochastic simulations*, the trains enter the system according to the timetable, with random delays generated for them. The decision as to whether a specific train is delayed or not is taken based on a generator of pseudorandom numbers obeying Bernoulli probability distribution (the values of the parameter are listed in Table 5). If a train is identified as delayed, a delay time is assigned to it. The delay times are generated based on the exponential probability distribution (the mean values are listed in Table 5). The random delay assignment to the trains follows the principles laid down in *SŽDC Directive SM124 – Railway Capacity Assessment* (*Směrnice SŽDC SM124 – Zjišťování kapacity dráhy*). No random effects other than the delays were applied in the simulation experiments.
- Each main replication is divided into 2 computation phases applying the same timetable as outlined earlier. The replication is started with a warming up period during which no data are collected for statistical processing. Instead, the system is filled with the potentially delayed trains. This stage is immediately followed by the main tracking period encompassing data collection for a post-simulation evaluation of the operational characteristics.
- Alternative station tracks and associated alternative line routes are allocated to the arriving trains when a conflict arises. Each train is associated with one primary route (containing the appropriate station track in line with the static track occupation plan) and three alternative routes using other station tracks (defined in the priority lists, shown in Table 6).

- A total of 100 replications of each main simulation (*mainReplCount* = 100) were accomplished in order to obtain an adequate range of simulation output data for statistical processing.
- One version of the system, with one specific timetable and one track layout, was investigated for this case study. This case study was processed by applying the methodical guidelines described in Section 4.
- The *maximum working (real) time* available for examinations (except the *probing phase*) associated with one version of the system during the addressing of a real project was postulated to be 200 hours. This is the time during which the appropriate simulation experiments must be completed and evaluated. It does not include activities required for setting up, verifying, and validating the target simulating system. The maximum time reserved for the pilot simulations was set at 100 hours (*pilotMaxLimit*) and another maximum of 100 hours was planned for simulations related to selected final experiments (*finalMaxLimit*). In addition, the maximum time reserved for the probing phase was set to 50 hours (*probingMaxLimit*). This means that the maximum time determined for the exploration of the given system version is 250 hours (*versionTimeLimit*).
- An overhead time ${}^{cost}t = 1$ minute was applied in the case study for evaluating the processing time estimates of the final experiments described next. This time was deliberately overestimated somewhat against the true values because the intent was to use rather pessimistic values of the overhead time in the estimates mentioned earlier.
- The RNS applications used the strategy of *hierarchically reduced lookahead duration*, with the primary lookahead duration being invariably set at 30 minutes (a period that realistically mirrors the operating lookahead of the dispatchers in actual rail traffic).

5.2 Characteristics of Decision-making Supports within MesoRail Tool

The value of the *Sum of Weighted Delay Increments* (*SWDI*) indicator encompassing all of the trains included (46 trains) is evaluated for each replication for optimization purposes. The delay increment of a train is represented by the difference between the delays at the input into and output from the simulating system. As mentioned earlier, train inputs and outputs occur always in the boundary stations (*East, West*, or *South*). Each train is connected with a weight reflecting its importance in the traffic. The weights were selected as follows in accordance with the *SŽDC Directive SM124*: $w_{Express} = 1.8$, $w_{Regio} = 1.0$, and $w_{Cargo} = 0.2$. When the life cycle of a train is over, the resulting increment is multiplied by the weight and the product is input to the *SWDI* computation. The case with the lowest *SWDI* value is rated best when comparing the different courses of traffic.

The traffic character for a simulation experiment E can be expressed through the statistical indicator *meanSWDI*, which is typically utilized as a *railway infrastructure capacity index* in connection with the use of simulation methods (*SŽDC Directive SM124*). Its value is calculated as the *arithmetic mean* of the SWDI-values of the replications belonging to the experiment E and is obtained through the function *CrOptim*. The 95% confidence interval is also calculated for the mean. The half-width of this interval can be expressed either as the absolute value (denoted as *halfWidth*) or by its relative value (referred to as *relatHalfWidth*), which is the ratio of the absolute value to the corresponding mean value:

$$relatHalfWidth = \frac{halfWidth}{meanSWDI} \tag{2}$$

The *meanSWDI* indicator is used to evaluate different conflict resolution variants; the variant/experiment possessing the lowest *meanSWDI* value is selected as optimal. This indicator is also used for the evaluation of the primary experiments consisting of main replications.

The statistical indicator *meanRelatHalfWidth* can be used for global evaluation of the attained precisions of the *meanSWDI* values (from multiple conflict resolution variants). This indicator is the mean of the *relatHalfWidth* values of all of the conflict resolution variants considered.

Note: Nested simulations are not used in simulation experiments for which $maxLevel = 0$: conflict resolution within the main replications are based on the sole use of the priority planning method.

5.3 Comparative Simulating System Developed in Villon Simulation Tool

To enable the results of certain computations within a mesoscopic simulating system (executed in the MesoRail tool) to be evaluated, a matching simulating system working on the microscopic level of detail was created within the Villon software tool. The system included the same track infrastructure, the same timetable, and the same input train delay generation strategy as MesoRail. Conflict resolution in Villon is based solely on priority planning using the same priority list as are those shown in Table 6.

Villon is routinely used to process simulation studies for big railway companies and has been successfully validated by the SŽDC with respect to the (i) train runtimes, (ii) train run tachograms, (iii) models of technological traffic processes, and (iv) performance of the interlocking systems.

Although either simulator works on its own level of detail, it appeared convenient to assess the movements of trains in MesoRail by coarse comparison with the corresponding movements in *Villon*.

Deterministic simulations using the specified timetable (Figure 8) were first performed in both simulation tools. The train runtimes were recorded for all trains during the simulations. The relative difference (*relDiff*) of the train runtimes was calculated for a comparison. The *relDiff* value for a specific train is calculated as the ratio of the absolute value of the difference between the train's runtimes in the MesoRail and Villon tools, respectively, to the lower of the two runtimes: $relDiff = (|T_{Meso} - T_{Villon}| / \min(T_{Meso}, T_{Villon})) \times 100\ [\%]$. The maximum relative difference for one train was 0.6 % and the mean relative difference for all of the trains simulated was 0.3%. Since the relative differences did not exceed the *a priori* determined boundary limit of 5%, it was concluded that the mesoscopic model in the MesoRail simulator works on the required level of precision. A comparison of the results collected from the *stochastic simulations* follows.

5.4 Computer Cluster Features

The computations of all simulation experiments within the case study were performed on a computer cluster having available a data repository (96 TB disk memory divided into 24 HDDs) and 4 computer nodes. Each node was equipped with (i) four Intel XEON 12-core processors, the *Hyper-Threading* (*HT*) *technology* enabling parallel execution of 96 threads; (ii) 384 GB RAM; and (iii) an SSD disc (with 1TB capacity) for the operating system. The cluster is interconnected via a 10 GB Ethernet network, serving for communication with the repository and mutually between the nodes. The CentOS Linux 7 operating system was run on the cluster.

5.5 Probing Phase

Selected probing experiments applying the RNS approach were first performed within the *probing phase* for the initial identification of the computational characteristics of the nested simulations. A number of different experiment configurations with different extents of nesting ($maxLevel \leq 3$) were applied and different numbers of nested replications ($nestReplCount \leq 20$) were used. The mean memory demand per main replication was found to be about 32 GB, although requirements

Table 7. Pilot Experiments Applying 16 Replications of the Main Simulations

Simulated period: **8 to 10 a.m.** (morning peak traffic) Number of LPUs: **16** Primary lookahead duration for the nested simulations: **30 minutes**										
	1	2	3	4	5	6	7	8	9	10
	max Level [−]	*nest Repl Count* [−]	*main Repl Count* [−]	*mean SWDI*± *halfWidth* (main) [min]	*mean Relat Half Width* (nest.) [−]	*mean RNS ConflCount* [−]	*mean ReplCount* [−]	$^{mean}t_{16}$ ± *halfWidth* (main) [h]	$^{real}T_{16}$ (main) [h]	$^{estim}T_{84}$ (main) [h]
Sc0000p	0	0	16	32.5 ± 7.0	–	0	1	0.10 ± 0.00	0.1	0.7
Sc0103p	1	3	16	26.3 ± 6.5	0.13	13	84	0.43 ± 0.06	0.7	3.2
Sc0105p	1	5	16	26.2 ± 6.5	0.08	12	138	0.47 ± 0.08	0.8	3.5
Sc0107p	1	7	16	26.2 ± 6.6	0.06	12	193	0.63 ± 0.11	1.1	4.7
Sc0110p	1	10	16	26.2 ± 6.5	0.05	12	276	0.84 ± 0.14	1.3	6.2
Sc0120p	1	20	16	26.2 ± 6.5	0.03	12	554	1.83 ± 0.26	3.1	13.2
Sc0203p	2	3	16	26.2 ± 6.5	0.08	224	1401	2.72 ± 0.75	5.2	21.8
Sc0205p	2	5	16	26.1 ± 6.6	0.05	363	3854	7.38 ± 2.51	20.7	62.0
Sc0207p	2	7	16	–	–	–	–	_	23.0 *	–
Sc0210p	2	10	16	–	–	–	–	_	–**	–
Sc0220p	2	20	16	–	–	–	–	_	–**	–
Sc0303p	3	3	16	–	–	–	–	_	23.0 *	–
Sc0305p	3	5	16	–	–	–	–	_	_**	–
Σ									79.0	

*The computations (related to 16 main replications) were not completed within 23 hours of the real-time (assumption: $^{estim}T_{84} > 70$ hours).

**The computations were not initiated (their durations were anticipated to be longer than the durations of the previous experiments applying the same values of the *maxLevel* parameter and the lower values of the *nestReplCount* parameter).

for allocation of up to 46 GB were also encountered in selected main replications. The total time consumed for completing the probing phase was 50 hours.

To enhance the efficiency of the computing operations, the computer cluster was divided (based on findings from the probing experiments) into LPUs. Each unit had available system tools adequate for the simulation of one main replication. Taking into account the peak demand for memory (46 GB per main replication) and the anticipated use of complex RNS configurations, 96 GB memory capacity was reserved for each main replication. To satisfy this requirement, the cluster was divided into 16 LPUs. Thus, each LPU had available 96 GB memory and one physical CPU (24 threads using HT technology). Hence, up to 16 main replications could be run in parallel on a cluster and each main replication can utilize 24 threads for parallel runs of the relevant nested simulations.

It holds for the computations associated with one specific final simulation experiment that each LPU is associated with one specific computation task represented by the set of replications $^{d}H_b\,(^{d}S_{0,0,r})$, where the parameters *r*, *d*, and *b* are fixed ($r \in \langle\ 1 \ldots 100\ \rangle$, $d \in \{0,1,2,3\}$, $b \in \{0,3,5,7,10,20\}$). This means that one computation task is represented by the computation of one main replication, including all of its hierarchically subordinated nested replications.

5.6 Pilot Phase

A total of 13 candidate scenarios were proposed for testing in this phase: one scenario (*Sc0000p*) was planned without the use of nested simulations and 12 scenarios were planned to use RNS

(Table 7). The structure of the scenario identifiers was *ScXXYYp*, where *XX* expressed the settings of the parameter *maxLevel* and *YY* reflects the value of the parameter *nestReplCount*.

Based on the logical division of the cluster into 16 LPU explained earlier, it was postulated for the pilot phase that the scenarios should include 16 replications of the main simulations (*mainReplCount* = 16).

Rough estimates of the time needed to complete the computation of the scenarios in the final phase, ${}^{estim}T_N$, were obtained by using Equation (1). Thus, the following settings are used in this study: $n = 16$, $N = 84$, $m = 16$, ${}^{cost}t = 1$ min. The ${}^{mean}t_{16}$ and ${}^{estim}T_{84}$ values for the different scenarios are listed in Table 7.

Before setting the *pilotMaxLimit* parameter, its impacts on the future final computations had to be considered. In this context, the *maximum duration in real time* (tentatively denoted ${}^{max}T_{84}$) admissible for future completion of the 84 main replications for each scenario in the subsequent *final phase* had to be set. Taking into account the 100 hours reserved for the whole final phase, the above time was set at ${}^{max}T_{84} = 70$ hours in the case study. This setting was driven by the requirement to enable potential execution of a maximum of one appreciably time-consuming simulation experiment following the appropriate scenario in the final phase. If the "maximalist" calculation is performed in reality, the remaining time of the final phase is potentially used for shorter RNS-based experiments and for the computation of one scenario in which RNS are not applied.

For the determination of the *pilotMaxLimit* parameter, the estimation of the real time required for the completion of the 84 replications (${}^{estim}T_{84}$) of the final scenario can be alternatively expressed as a function depending on the estimate of the duration of the appropriate pilot computation (${}^{estim}T_{16}$) executing 16 replications. If the time ${}^{repl}t$ is the same in the general calculation of the two estimates, then the function can be expressed from Equation (1) as:

$$ {}^{estim}T_{84} = 3.125 \times {}^{estim}T_{16} \tag{3} $$

Since ${}^{estim}T_{84} \leq {}^{max}T_{84}$ (${}^{max}T_{84} = 70$ hours), Equation (3) can be rearranged to obtain the pilot calculation duration estimate as follows: ${}^{estim}T_{16} \leq (0.32 \times {}^{max}T_{84}) = 22.4$ hours. In this context, the *pilotMaxLimit* parameter was set at 23 hours.

Table 7 lists the processed results of the pilot experiments. Columns 1 and 2 include the maximum permitted numbers of levels of nesting (*maxLevel* parameter) and the numbers of nested replications applied to the testing of the conflict resolution variants (*nestReplCount* parameter). Column 4 lists the main simulation results: the *meanSWDI* indicator values, including the half-widths of the 95% confidence intervals (*halfWidth*). The calculation of the data in this column was based on the processing of the data from the main replications. Column 5 displays data of the *meanRelatHalfWidth* indicator, which is the outcome of the processing of data obtained from all nested replications for the appropriate scenarios. The data document the mean relative precision of determination of the *meanSWDI* indicator (obtained from the nested simulations) for different scenarios, particularly for different settings of the *nestReplCount* parameter.

The next two columns represent the mean numbers of conflicts that were addressed via RNS (*meanRNSConflCount*) and the mean numbers of replications (*meanReplCount*) for the sets ${}^{d}H_b({}^{d}S_{0,0,r})$, where $r = 1 \ldots 16$ and the superscript d and subscript b for each scenario match the values in columns 1 and 2. Scenario *Sc0205p* can serve as an example where the sets ${}^{2}H_5({}^{2}S_{0,0,r})$, $r = 1 \ldots 16$ contained an average of 3854 replications and 363 conflicts addressed via RNS.

Column 8 contains information on the mean duration of one main replication (${}^{mean}t_{16}$), including the half-widths of the 95% confidence intervals. The last two columns display information on the observed global periods of the real time (${}^{real}T_{16}$) required to perform the pilot experiments and on the estimates of the expected total periods of the real time (${}^{estim}T_{84}$) required to complete the final experiments (84 replications).

The results in Table 7 document that the pilot experiment computations were fully completed for 8 scenarios while for 2 scenarios the computations were prematurely terminated because the time allocated had been spent before completing the experiments. For 3 proposed scenarios, the computations were not even initiated with regard to the fact that the preceding scenarios with the same number of levels of nesting (*maxLevel*) and a lower number of nested replications (*nestReplCount*) had been too time-consuming. The total time required for completing the pilot phase was 79 hours.

The results demonstrate that the use of scenarios applying 3 levels of nesting is unfeasible because the computation lasts too long, exceeding the specified time frame.

Table 9 includes the results of the completed simulation experiments using those scenarios for which the pilot computations had been prematurely terminated when applying the rules described. Such computations would not be made in a real project and are presented here by way of illustration.

5.7 Final Phase

Once the pilot experiments were over, the scenarios *Sc0120p* and *Sc0205p* were selected for completion. The *meanSWDI* indicator values, which are calculated from the data of the main replications, were nearly identical (including the *halfWidth* values) for all of the completed pilot experiments that applied RNS. Therefore, lower values of the *meanRelatHalfWidth* indicator (mirroring the results of computations of the nested simulations examining the conflict resolution variants) were given preference when selecting the above-mentioned scenarios, and the estimated final experiment completion times (${}^{estim}T_{84}$) were taken into consideration. The scenario *Sc0000p* was completed because this offered the opportunity to compare the results between the scenarios applying RNS and scenarios in which the nested simulations approach is not applied.

The *meanSWDI* indicator value was around 25 minutes for the scenarios *Sc0120* and *Sc0205* using RNS and approximately 33 minutes for the scenario (*Sc0000*) in which nested simulations are not used (Table 8). Seen from this perspective, the application of RNS for decision-making support provided up to 25% better results compared with simulations in which no RNS were used, applying only static priority lists for conflict resolution.

By way of illustration, Table 8 also includes the results of scenarios that were not selected for completion after the pilot phase but were not classed as unsuitable with respect to the duration of the final calculations. The results would not be completed in a real project.

Sixty-nine hours were spent on the final experiments (*Sc0000, Sc0120* and *Sc0205*), preceded by 76 hours of pilot experiment calculations and 50 hours of probing experiments. Thus, the total time of 250 hours was not exceeded.

If a time frame different from the initial 200 hours is postulated for testing one version of the system, then different scenarios for testing in the pilot phase must be devised based on the new specific time frame. Subsequently, a different criterion must be applied to the selection of the scenarios to be used for execution completion in the final phase. For instance, if the time frame is significantly shorter than the 200 hours, then the final phase will apparently use a scenario (scenarios) applying no more than one level of nesting (i.e., the setting would be $maxLevel = 1$).

5.8 Evaluation of the Results

Beyond the framework of the final phase, the results of the experiments described here were assessed by comparison with those of the simulations in the well-established Villon simulator. As mentioned in Section 4.3, the simulating system configuration was identical to that in the MesoRail tool. Priority lists were used for decision support in the stochastic simulations. The

Table 8. Results of the Complete Experiments Applying 100 Main Replications

Simulated period: **8 to 10 a.m.** (morning peak traffic)								Number of LPUs: **16**		
Primary lookahead duration for nested simulations: **30 minutes**										
	1	2	3	4	5	6	7	8	9	10
	max Level [−]	*nest Repl Count* [−]	*main Repl Count* [−]	*mean SWDI ± halfWidth* (main) [min]	*mean Relat Half Width* (nest.) [−]	*mean RNS ConflCount* [−]	*mean ReplCount* [−]	${}^{real}T_{84}$ (main) [h]	${}^{estim}T_{84}$ (main) [h]	${}^{real}T_{16}$ (main) [h]
Sc0000	0	0	100	32.6 ± 3.2	–	0	1	0.7	0.7	0.1
Sc0120	1	20	100	25.2 ± 2.5	0.03	13	567	11.9	13.2	3.1
Sc0205	2	5	100	24.6 ± 2.5	0.05	386	4120	56.5	62.0	20.7
Σ								69.1		
Sc0103	1	3	100	25.4 ± 2.6	0.13	13	84	2.7	3.2	0.7
Sc0105	1	5	100	25.2 ± 2.5	0.08	13	140	3.2	3.5	0.8
Sc0107	1	7	100	25.2 ± 2.5	0.06	13	197	4.1	4.7	1.1
Sc0110	1	10	100	25.3 ± 2.6	0.05	13	283	6.2	6.2	1.3
Sc0203	2	3	100	24.9 ± 2.6	0.08	236	1504	24.2	21.8	5.2

Table 9. Illustration of the "Non-promising" Experiments from Table 7

Simulated period: **8 to 10 a.m.** (morning peak traffic)								Number of LPUs: **16**		
Primary lookahead duration for the nested simulations: **30 minutes**										
	1	2	3	4	5	6	7	8	9	10
	m L C [−]	*n R C* [−]	*m R C* [−]	*mean SWDI ± halfWidth* (main) [min]	*mean Relat Half Width* (nest.) [−]	*mean RNS ConflCount* [−]	*mean Repl Count* [−]	${}^{mean}t_{16} \pm$ *halfWidth* (main) [h]	${}^{real}T_{16}$ (main) [h]	${}^{estim}T_{84}$ (main) [h]
Sc0207p	2	7	16	26.1 ± 6.6	0.04	490	7214	17.9 ± 10.1	80.0	**174.9**
Sc0210p	2	10	16	26.0 ± 6.6	0.03	704	14791	34.6 ± 13.1	91.8	**216.2**
Sc0220p	2	20	16	26.0 ± 6.6	0.02	1394	59085	185.0 ± 68.3	560.6	**1583.4**
Sc0303p	3	3	16	26.2 ± 6.6	0.06	2881	18010	31.8 ± 20.1	127.2	**324.4**
Sc0305p	3	5	16	26.0 ± 6.6	0.04	7315	74670	185.6 ± 116.8	741.8	**1890.1**

meanSWDI ± halfWidth indicator obtained after 100 replications was 32.1 ± 3.7 minutes. The difference in the *meanSWDI* indicator between this computation and the computation for the scenario *Sc0000* in the MesoRail tool was lower than 2%, demonstrating a very good precision of the MesoRail computations. A comparison with the results of the scenarios *Sc0120* and *Sc0205* gives evidence of superiority of the optimization approach in MesoRail (applying *RNS*), providing a value of the *meanSWDI* indicator roughly 22% better than the model that was set up in Villon.

These facts give evidence that it is worthwhile to make efforts to additionally improve the modelling capabilities of railway traffic simulators, for example, by using the RNS concept as a decision-making support.

Therefore, where the throughput/capacity of the infrastructure of a passenger railway station is examined, the RNS approach can be used advantageously in simulations testing diverse versions

of the system analyzed. Based on the results of simulations of various traffic versions, the versions can be compared and sorted by the operational characteristics obtained.

6 CONCLUSIONS

Sequential railway traffic simulations currently constitute a routine approach to the examination and optimization of existing as well as planned railway systems. In this context, considerable attention is paid to the development of novel approaches and methods contributing to quality improvements of the modelling options and optimization techniques for the applicable simulators.

The method of reflective nested simulations is one of the feasible optimization approaches to automated decision-making support in the domain of operational problems arising during traffic simulations. The case study presented here used nested simulations in support of decision-making associated with the resolution of conflicts regarding the allocation of platform tracks in a passenger railway station. The use of this method appeared to be successful: the nested simulations concept proved to have a better optimization potential than the static priority planning approach routinely used in existing simulators.

The new theoretical and practical results of the R&D activities described in this article include the following:

- *Framework guidelines for the use of nested simulations within application projects* where specific time limitations for their executions exist. In this approach, the life cycle of the experimental stage of examination of one system version is divided into two principal phases. The *pilot phase* includes computation of relative low numbers of main replications for the diverse simulation experiment scenarios. Once the time demands of the computations are known, the durations of the final computations (i.e., computations encompassing the full planned number of main replications) are estimated by using expertise in the technical solution of the computation parallelization issue (regarding sequential simulators applying RNS). The estimates are used, in conjunction with the applicable criterion, to select suitable scenarios for the final phase.
- The specific *control algorithm* that *synchronizes the parallel executions of independent replications* related to sequential simulators applying RNS. The algorithm utilizes a model of a *dynamic rooted tree* whose leaves reflect all (sequential) replications that can be computed in parallel.
- *Hierarchically reduced lookahead durations* used within nested simulations. This lookahead concept is very suitable for application in the rail traffic domain, where it quite naturally reflects the traffic forecasting information that is available to the dispatchers.
- *Technical solution for implementation of the parallel computations on a computer cluster* or another computing system, which can be realized through LPUs. Such units make parallel computations of the main replications. The parallel computations of nested replications belonging to one main replication and associated with the testing of different solution variants for the operational problems addressed are always performed on one processing unit by applying the multi-thread approach.
- *MesoRail software,* a tool designed for railway traffic simulations on a mesoscopic level of detail. This tool was used for the implementation of the conceptual approaches and for supporting computations on computer clusters and on PCs.

Like in the MesoRail tool, the nested simulations method can be incorporated into existing well-established (commercial) simulation tools, which must be additionally modified for the

implementation of the experiments on computer clusters. This expansion of the tools is one of the ways to improve their optimization capabilities.

If the nested simulations method is implemented on a PC, then the main replications are computed sequentially. The computation time for one main replication can approach that on a cluster within one logical processing unit (depending on the CPU type in the PC and on the number of computing threads available for the computations of the nested replications). The main advantage of the computer cluster (when using the parallelization method described here) is the possibility of using computations that run N main replications in parallel, which results in a roughly N-fold acceleration of the computation process compared with a conventional one-processor PC.

The application of reflective nested simulations is beneficial particularly when a not too large number of acceptable variants must be considered in solving a decision-making problem. The method is generally applicable mainly in support of medium-term (tactical) and long-term (strategic) planning of not only railway traffic but also of other system types, the analysis of which uses simulators working on the microscopic, mesoscopic, or macroscopic level of detail. Nested simulations are not recommended for support of short-term (operative) planning or control because of the high demands put on the computational capabilities and excessive time demands.

Given the high computation demands of the reflective nested simulations, the focus can be on a deep analysis of the options available for reducing the requirements for memory capacity and computation time of simulations in question. Specifically, main efforts are expected to be aimed at reductions of the hierarchically ordered sets of nested replications. The reductions are supposed to be based (i) on substitution solutions for selected conflicts (using "non-simulation" methods requiring less demanding computations) and (ii) on the elimination of whole selected replications that have been identified as non-promising by the heuristic approach. The aim is to find approaches that will result in a reduction of the computational demands and will be associated with a favorable relation between the acceleration of the nested simulations and any overall lower of quality of result. However, it is theorized that these approaches will not be generally applicable but that they will meet the needs of a particular system.

REFERENCES

[1] N. Adamko and V. Klima. 2008. Optimisation of railway terminal design and operations using Villon generic simulation model. *Transport* 23, 4 (2008), 335–340. https://doi.org/10.3846/1648-4142.2008.23.335-340

[2] J.-P. Bendfeldt, U. Mohr, L. Muller, and A. RailSys. 2000. System to plan future railway needs. *WIT Transactions on Built Environment* 50 (2000), 249–255. https://doi.org/10.2495/CR000241

[3] A. Nash and D. Huerlimann. 2004. Railroad simulation using opentrack. *WIT Trans. Built Environ* 74 (2004). https://doi.org/10.2495/CR040051

[4] M. Zhong, Y. Yue, and D. Li. 2019. Analyzing and evaluating infrastructure capacity of railway passenger station by mesoscopic simulation method. In *2018 International Conference on Intelligent Rail Transportation, ICIRT 2018*, Institute of Electrical and Electronics Engineers Inc., Singapore, 1–5. DOI: https://doi.org/10.1109/ICIRT.2018.8641593

[5] M. Marinov and J. Viegas. 2011. A mesoscopic simulation modelling methodology for analyzing and evaluating freight train operations in a rail network. *Simulation Modelling Practice and Theory* 19, 1 (2011), 516–539. Retrieved November 12, 2019 from www.arpnjournals.com.

[6] NEMO—Netz-Evaluations-Modell bei den ÖBB | Eurailpress Archiv, (n.d.). Retrieved November 12, 2019 from https://eurailpress-archiv.de/SingleView.aspx?lng=en&show=16622.

[7] C. F. Tan, L. S. Wahidin, S. N. Khalil, N. Tamaldin, J. Hu, G. W. and M. Rauterberg. 2019. The application of expert system: A review of research and applications. *Journal of Engineering and Applied Sciences* 11, 4 (2016), 2448–2453. Retrieved November 12, 2019 from www.arpnjournals.com.

[8] Alexander Fay and E. Schnieder. 1999. Knowledge-Based Decision Support System for Real-Time Train Traffic Control. In: N. H. M. Wilson (Ed.), *Computer-Aided Transit Scheduling*. Springer, Berlin, 347–370.

[9] A. Fay. 2000. Fuzzy knowledge-based system for railway traffic control. *Engineering Application of Artificial Intelligence* 13, 6 (2000), 719–729. https://doi.org/10.1016/S0952-1976(00)00027-0

[10] K. Komaya and T. Fukuda. 1990. ESTRAC-III: An expert system for train traffic control in disturbed situations. In *IFAC Proceedings* 23, 2 (1990), 147–153. https://doi.org/10.1016/S1474-6670(17)52664-6

[11] K. Komaya. 1991. A new simulation method and its application to knowledge-based systems for railway scheduling. In *Proceedings of the 1991 ASME/IEEE Joint Conference on Rail.* IEEE, 59–66. https://doi.org/10.1109/rrcon.1991.160928

[12] Y. Larroche, R. Moulin, and D. Gauyacq. 1994. Sepia: A real-time expert system that automates train route management. In *IFAC Proceedings* 27, 12 (1994), 977–982. https://doi.org/10.1016/S1474-6670(17)47601-4

[13] W. Deng, H. Zhao, and H. He. 2018. Research on heavy haul railway dispatching system based on fuzzy expert system. In *DEStech Transactions on Computer Science and Engineering* CCNT (August 2018), 317–323. https://doi.org/10.12783/dtcse/ccnt2018/24719

[14] Y.-H. Cheng and L.-A. Yang. 2009. A fuzzy petri nets approach for railway traffic control in case of abnormality: Evidence from Taiwan railway system. *Expert Systems with Applications* 36, 4 (2009), 8040–8048. https://doi.org/10.1016/J.ESWA.2008.10.070

[15] C. R. Standridge and D. Steward. 2000. Using expert systems for simulation modeling of patient scheduling. *Simulation* 75, 3 (2000), 148–156. https://doi.org/10.1177/003754970007500303

[16] L. Bai, T. Bourdeaud'huy, B. Rabenasolo, and E. Castelain. 2014. A mixed-integer linear program for routing and scheduling trains through a railway station. In *Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems.* SCITEPRESS—Science and Technology Publications, 445–452. https://doi.org/10.5220/0004863104450452

[17] P. Chakroborty and D. Vikram. 2008. Optimum assignment of trains to platforms under partial schedule compliance. *Transportation Research Part B Methodological* 42, 2 (2008), 169–184. https://doi.org/10.1016/j.trb.2007.07.003

[18] A. Azadeh, S. F. Ghaderi, and H. Izadbakhsh. 2008. Integration of DEA and AHP with computer simulation for railway system improvement and optimization. *Applied Mathematics and Computation* 195, 2 (2008), 775–785. https://doi.org/10.1016/j.amc.2007.05.023

[19] D. De Luca Cardillo and N. Mione. 2017. k L-list λ colouring of graphs. *European Journal of Operational Research* 106, 1 (1998), 160–164. https://doi.org/10.1016/S0377-2217(98)00299-9

[20] M. Bažant and A. Kavička. 2009. Artificial neural network as a support of platform track assignment within simulation models reflecting passenger railway stations. *Proceedings of the Institution of Mechnical Engineers, Part F: Journal of Rail and Rapid Transit* 223, 5 (2009), 505–515. https://doi.org/10.1243/09544097JRRT268

[21] I. Krivý and E. Kindler. 2006. Terminology of nested simulation models. In *International Conference on Computer Systems and Technologies (CompSysTech'06).* IIIA.5-1–IIIA.5-6.

[22] E. Kindler. 2010. Nested models implemented in nested theories. In *Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling and Simulation 2010,* 150–159.

[23] E. Kindler. 2006. Nesting simulating agents in SIMULA. In *20th European Conference on Modelling and Simulation, European Council for Modeling and Simulation, Bonn.* 526–531. DOI: https://doi.org/10.7148/2006-0526

[24] J. B. Gilmer and F. J. Sullivan. 1999. Multitrajectory simulation performance for varying scenario sizes [combat simulation]. In *Winter Simulation Conference Proceedings.* Phoenix, AZ, IEEE, 1137–1146. https://doi.org/10.1109/wsc.1999.816832

[25] J. B. Gilmer and F. J. Sullivan. 2000. Recursive simulation to aid models of decisionmaking. In *Proceedings of the Winter Simulation Conference.* IEEE, Orlando, FL, USA, 958–963. DOI: https://doi.org/10.1109/wsc.2000.899898

[26] B. Bonté, R. Duboz, G. Quesnel, and J.-P. Müller. 2009. Recursive simulation and experimental frame for multiscale simulation. In *Proceedings of the 2009 Summer Computer Simulation Conference, Society for Modeling & Simulation International, Istanbul, Turkey.* 164–172. DOI: https://doi.org/10.13140/2.1.3364.3521

[27] M. B. Gordy and S. Juneja. 2010. Nested simulation in portfolio risk measurement. *Management Science* 56, 10 (2010), 1833–1848. https://doi.org/10.1287/mnsc.1100.1213

[28] M. Hybinette and R. M. Fujimoto. 2001. Cloning parallel simulations. *ACM Transactions on Modeling and Computer Simulation* 11, 4 (2001), 378–407. https://doi.org/10.1145/508366.508370

[29] X. Li, W. Cai, and S. J. Turner. 2017. Cloning agent-based simulation. *ACM Transactions on Modeling and Computer Simulation* 27, 2 (2017), 1–24. https://doi.org/10.1145/3013529

[30] P. Pecher, J. Crittenden, Z. Lu, and R. Fujimoto. 2018. Granular cloning: Intra-object parallelism in ensemble studies. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'18).* ACM, 165–176. https://doi.org/10.1145/3200921.3200927

[31] A. Pellegrini, R. Vitali, and F. Quaglia. 2015. Autonomic state management for optimistic simulation platforms. *IEEE Transactions on Parallel Distributed Systems* 26, 6 (2015), 1560–1569. https://doi.org/10.1109/TPDS.2014.2323967

[32] M. Principe, T. Tocci, P. Di Sanzo, F. Quaglia, and A. Pellegrini. 2020. A distributed shared memory middleware for speculative parallel discrete event simulation. *ACM Transactions on Modeling and Computer Simulation* 30, 2 (2020). https://doi.org/10.1145/3373335

[33] J. S. Lee, S. Andradóttir, and R. M. Fujimoto. 2017. Simulation cloning with induced negative correlation. *Journal of Simulation* 11, 4 (2017), 391–406. https://doi.org/10.1057/s41273-016-0028-7

[34] R. Diviš and A. Kavička. 2016. The method of nested simulations supporting decision-making process within a mesoscopic railway simulator. In *28th European Modeling and Simulation Symposium (EMSS'16), Dime University of Genoa, Larnaca, Cyprus.* 100–106.

[35] R. Diviš and A. Kavička. 2015. Design and development of a mesoscopic simulator specialized in investigating capacities of railway nodes. In *27th European Modeling and Simulation Symposium (EMSS'15), Dime University of Genoa, Bergeggi, Italy.* 52–57.

[36] R. Diviš and A. Kavička. 2019. Computational optimizations of nested simulations utilized for decision-making support. In *31st European Modeling and Simulation Symposium (EMSS'19), Dime University of Genoa, Lisbon, Portugal.* 80–89.

Received February 2020; revised February 2021; accepted May 2021