DOI: https://doi.org/10.46354/i3m.2019.emss.014

COMPUTATIONAL OPTIMIZATIONS OF NESTED SIMULATIONS UTILIZED FOR DECISION-MAKING SUPPORT

Roman Diviš[a], Antonín Kavička[b]

[a],[b]University of Pardubice, Faculty of Electrical Engineering and Informatics

[a]roman.divis@upce.cz, [b]antonin.kavicka@upce.cz

ABSTRACT
Nested simulations present a general method suitable for use in realizing a multi-trajectory simulation or as a decision support in a simulator. The principle of nested simulation (as a decision support) is to find a solution to a problem using other time-limited simulations which verify alternative options. After the nested simulations have finished, the solutions of individual alternatives are assessed and the best solution is applied to the main simulation.
The aim of this article is to test various exact and heuristic approaches to reduce the computational complexity of nested multi-simulations. The basic method uses the recorded states from previous simulations to avoid duplicate calculations leading to the same states. Other measures are heuristic and are based on the detection of similar states/replications during the calculation.

Keywords: nested simulations, decision support, rail transport

1. INTRODUCTION
The stochastic simulator requires implementation of some of the decision support techniques. There are a number of ways to solve conflicting situations - from leaving the solution to the user (interactive mode of simulation) through various complex mathematical models (fuzzy logic, artificial neural networks, ...). Even though there are a number of complex decision support techniques, in practice, one of the easiest methods - priority planning - is often implemented in microscopic railway simulators. This method is used in OpenTrack, Villon and other rail simulators.
Nested or recursive simulations represent another methodology that can be used to support decision making in simulators. The principle of the method consists in suspending the actual main simulation at the moment when the conflict situation occurs and the simulation is then cloned into several variants. Individual clones have modified parameterization to allow testing of different options for resolving the conflict situation. These nested/recursive simulations (basically different outlooks for the future within a limited time horizon) are triggered, and after a certain period of time, it is evaluated which one has the best results. The optimal one is then used as a solution and the main simulation continues only with the chosen variant. Nested simulations are implemented and tested as experimental decision support in the developed MesoRail simulation tool (Diviš and Kavička 2015).
The aim of this paper is to describe possible exact and heuristic algorithms designed to optimize nested simulation calculations. The aim of each method is to reduce the machine time required for simulation experiments. The proposed methods use different techniques based mainly on recording reached states during simulations to speed up future calculations.

1.1. Brief overview of the state-of-the-art
It has to be declared that not many authors pay attention to the research of nested/recursive simulations.
The authors Gilmer and Sullivan were focused in several of their articles on the efficiency of higher number of replications in contrast with multi-trajectory simulation (Gilmer and Sullivan 1999). Their main interest is related to the military simulator Eaglet, which simulates the movement of military units of two armies and their mutual interactions.
Eugen Kindler (as a pioneer of nested simulation in Europe) published many articles with the focus on both, the theoretical description of nested simulations (classification, terminology, etc.) and their applications in practice (Kindler 2010).
The issue of a planning support system is discussed by Hill, Surdu, Ragsdale, and Schafer (2000). Those authors were engaged in military planning.
Another area of applied nested simulations is connected with scalable simulation models, which allow applying both a macroscopic and a microscopic level of investigation within the frame of one simulator (Bonté, Duboz, Quesnel and Muller 2009). Another area of exploiting nested simulations is financial a risk management – e.g. Gordy and June (2010).

2. POSSIBLE APPROACHES TO THE USE OF NESTED SIMULATIONS
The method of *nested/recursive simulations* presents a principle of a simulation inside a simulation used to examine the results of multiple alternative scenarios (or developments) of the simulation. One possible use of this method is *decision support in a simulation*. Nested simulations run for a limited time, and after they are completed, their results are evaluated. Subsequently, the

nested simulations are merged back into a single instance and the main simulation can continue in a selected manner. The flow chart of a simulation using decision support with the use of nested simulations is shown in figure 1.

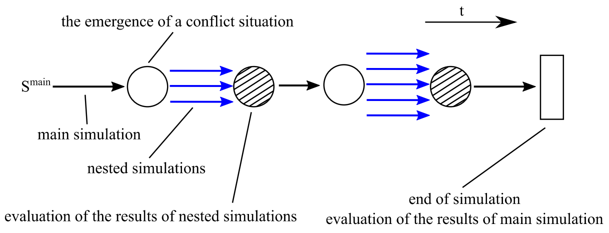

the emergence of a conflict situation
t
S^main
main simulation
nested simulations
end of simulation
evaluation of the results of nested simulations
evaluation of the results of main simulation

Figure 1: Using nested simulations as simulation decision support technique

Another use of nested simulations can be implemented as a *multi-trajectory simulation* - the simulation experiment is divided into nested simulations at individual points of decision and the simulation is gradually branched more and more and various scenarios are explored. According to the article (Gilmer and Sullivan 1999), this process can be more effective than using a large number of replications of a single simulation scenario. The flow chart of a simulation using a multi-trajectory simulation is shown in figure 2. In addition, attention will be paid to the first method of use – as a decision support in simulation.

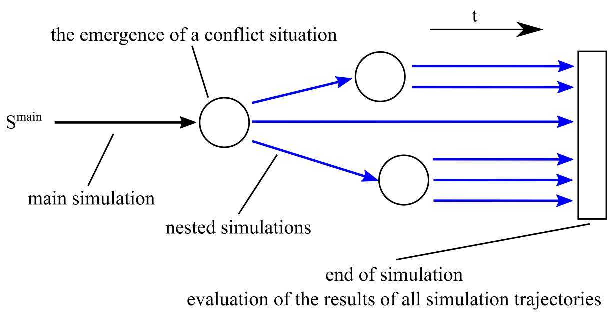

the emergence of a conflict situation
t
S^main
main simulation
nested simulations
end of simulation
evaluation of the results of all simulation trajectories

Figure 2: Using nested simulations to implement multi-trajectory simulation

3. NESTED SIMULATION TECHNIQUE

Decision support in the simulator can use the standard simulation engine for both the main simulation and nested simulations.

The method of using nested simulations as a support for decision making is not trivial and its detailed description can be found in the article (Diviš and Kavička 2016). The basic principle consists in pausing the execution of the main simulation at the occurrence of conflict (decision point of simulation). Subsequently, possible variants of the solution are identified, nested replications generated for each variant, and these replications run for a limited time (simulation lookahead). Upon completion, the simulation results are evaluated, and the main simulation continues with the selected solution. The strategy of using nested simulations is shown in figure 3. The whole technique of nested simulations is quite complicated, and for a better understanding of the procedures, several basic functions can be described:

- *simulation* - realizes the progress of one particular simulated replication,
- *solveConflict* - the function is called at the moment of the conflict emergence and decides how to solve it (using nested simulations or other methods),
- *solveUsingNestedSimulations* - performs all necessary steps to create nested simulations, calculate them and collect results,
- *createReplications* - creates specific replications for each possible variant of conflict solution.

The following are the formalizations (using pseudocode) of the above functions. A certain degree of abstraction was chosen with regard to the appropriate demonstration of individual methods leading to a reduction in the computational complexity of the nested simulation method.

```
function simulation(s):
while s.isNotEnd():
    e := s.nextEvent()
    s.processEvent(e)

    if s.isConflictOccured():
        solution := solveConflict(s, s.getConflict())
        s.applySolution(solution)

    s.saveState()
```

```
function solveConflict(s, c):
if simulation.recursionLevel <= REC_LIMIT:
    return solveUsingNestedSimulations(s, c)
else:
    return solveUsingAlternativeMethod(s, c)
```

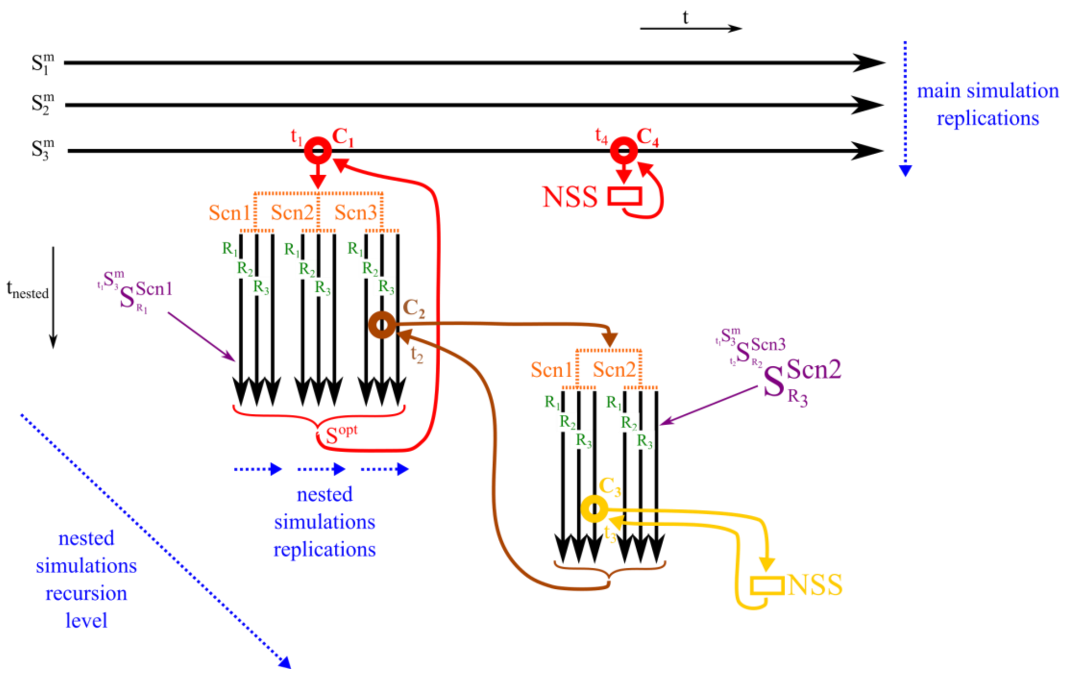
```
function solveUsingNestedSimulations(s, c):
variants := generateSolutionVariants(s, c)
seeds := generateRandomSeeds(s, REPL_COUNT)
nestedSimulations := createReplications(variants, seeds)

parallel start all nestedSimulations
wait until all finished nestedSimulations

solution := analyseResults(nestedSimulations)
return solution
```

```
function createReplications(variants, seeds):
simulations = {}
for variant in variants:
    for seed in seeds:
        s := createSimulation(baseSimulation, variant,
            seed)
        append s to simulations[variant]

return simulations
```



t
main simulation replications
NSS
Scn1 Scn2 Scn3
t_{nested}
S^{opt}
nested simulations replications
nested simulations recursion level
NSS

Figure 3: Illustration of occurrence and process of solving one conflict situation in one simulation replication

3.1. Basic parameters of the nested simulations

Before performing the nested simulations, we need to set the parameter group. This set of parameters may vary according to the type of conflict situations to be addressed. Depending on the implementation of decision support using nested simulations, the set of parameters may vary, or some parameters may be implemented differently.

Basic parameter overview:

- *ScnCount* – limit the number of evaluated scenarios (variants),
- *ScnGen* – generator of possible scenarios for solving conflict situation,
- *StopCond* – condition for terminating execution of nested simulations (can be interpreted as a constant value – length of lookahead),
- *CrOpt*, *CrOptComparer* – the functions used to evaluate individual nested simulation results.

Parameters related to multiple replications:

- *ReplCount* – number of evaluated multiple replications for each scenario (variant),
- *ReplResultsAggregateFunction* – function for evaluating results between replications of one scenario.

Parameters related to recursive nested simulations:

- *RecLimit* – maximum depth of recursion of nested simulation calculations,
- *RecStopCondBehaviour* – determines the use of *StopCond* in recursive simulations (extension of lookahead, limitation of lookahead to the original value, ...),
- *RecOnLimitAction* – defines the method (process) for resolving the conflict arising at the maximum recursion depth (no more recursive (nested) simulations are allowed).

More detailed description of individual parameters can be found in the article (Diviš and Kavička 2018).

4. EXACT OPTIMIZATION METHOD OF RECURSIVE CALCULATIONS

A series of simulations are needed to achieve relevant results in the simulation study. Individual simulations differ in the seeds used for pseudorandom number generators and thus affect the run of simulations. During the past investigation of nested simulation behaviour, it was found that the maximum nesting level dominates the quality of the result. A higher nesting level allows for better results. The negative consequence of allowing a high nesting level is a significant increase in the processing complexity of the method. If there is no initial estimate, it is not advisable to run simulations with a higher nesting level. There might also be a situation where the configuration is not computable with the available computing resources (too high computational time and/or memory requirements).
An iterative approach to a simulation study can start from the simplest of configurations and continue with more complicated configurations. If recursion is not allowed in the decision-making process at the beginning, the simulations can be completed very quickly. In the next iteration step, one level of recursion would be allowed and computational complexity would increase as expected. This procedure offers a safe way

to get results for all the necessary simulations, even if the available time to perform the study expires at the arbitrary nesting level. The disadvantage of the method is a considerable amount of repetition of previously performed simulations.
As part of our investigation (a case study on rail traffic simulation), it turns out that results do not change significantly between nesting levels. More levels of nesting rather refine decision-making results. It is not yet possible to generalize this behaviour due to the absence of multiple case studies. But for an application class where nested simulations exhibit the above behaviour, it is possible to continually save the progress of the calculation to an external memory (hard disk). Then these simulations are used to speed up calculations of new simulations with a higher nesting level. A more detailed level of simulations will use the capability to skip over already calculated sets of states. These jumps in (simulation) time can be used if a conflict is found whose new solution is identical to the solution from the previous set of simulations. This procedure can be repeated for each partial conflict as long as the simulation progress is the same, and there is no difference between the calculated solutions of the simulation. Each simulation branch (main simulation and individual nested simulations) represents an independent process to which this procedure can be applied.
For a better understanding of the method and the solved situations, the whole simulation strategy is illustrated in the following figures. Figure 4 (case A) shows the basic run of a stochastic simulation where a conflict situation occurred. The depicted situation does not allow nested simulations to be used, and thus conflict is resolved using an alternative method (e.g. priority planning). The selected variant of the solution is represented by a blue text label.

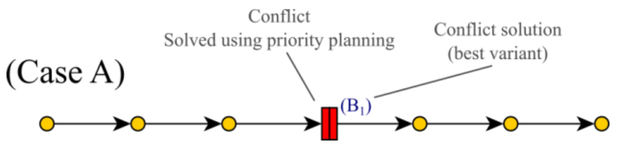


Figure 4: Conflict solved using priority planning (A)

Figure 5 (case B) shows the same simulation with one level of nested simulations allowed. The decision process of nested simulations is shown here as purple rectangles and the solution found is shown in a text label.

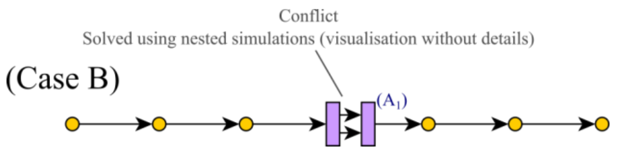
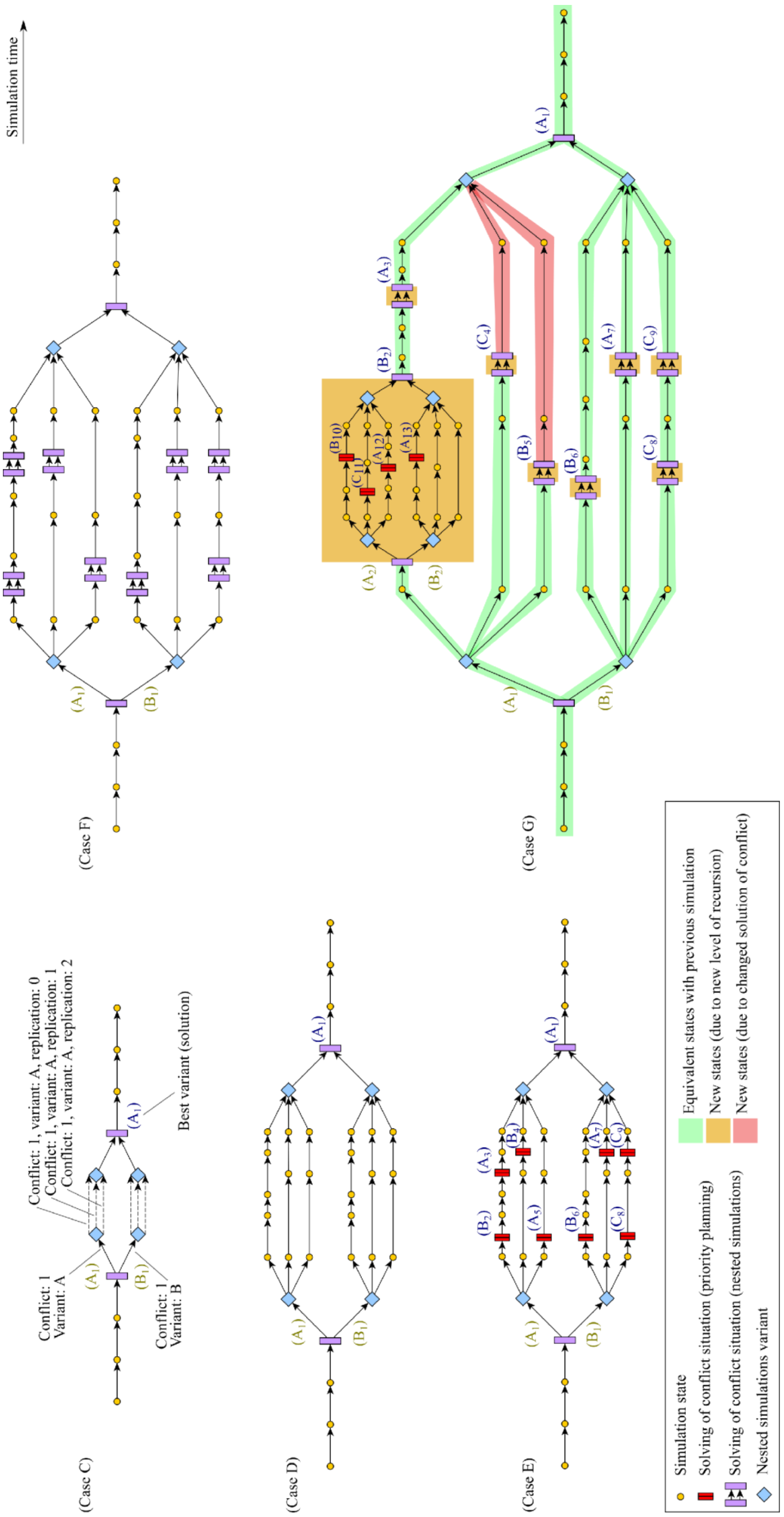

Conflict
Solved using nested simulations (visualisation without details)
(Case B)
(A1)

Figure 5: Conflict solved using nested simulations (B)

Figure 6 (case C) extends the previous representation to indicate solution variants that are evaluated by nested simulation. In this illustration, however, the details of the progress of the individual replications of nested simulation variants are omitted.
Figure. 6 (case D) adds a detailed view of each replication in nested simulations. In this representation, the progress of all nested replications is conflict-free and there is no subsequent nested conflict.
Figure 6 (case E) alternates the previous simulation and expands it with possible conflict situations within nested replications. Nested simulations are not allowed to be used as a solution for nested conflicts, but apply the priority planning method. This representation contains all the simulation details that were triggered with the maximum nesting level of 1.
Figure 6 (case F) is based on simulation case E and increases the maximum nesting level to 2. The original nested conflicts are not resolved by priority planning, but by nested simulations.
Figure 6 (case G) completes a detailed view of solving one selected nested conflict. Furthermore, the states and parts of the simulation are highlighted in green, orange, or red. The green colour is used for states that can be skipped (known states from the previous level of simulation; skip is possible to the last state in the green field). These states were already evaluated using a lower nesting level and they are identical to states in the new simulation (with higher nesting level). Orange states are new and related to a higher nesting level. These states need to be newly calculated; data is not available. The red-highlighted states represent changes from the previous simulation. The changes occurred due to the selection of a different solution (variant) to a conflict. These states need to be recalculated, and the corresponding data from the previous simulation cannot be used.
The example shows several possible situations that may occur during the simulation. If the new nesting level provides the same solution as the previous level of simulations - it is possible to effectively skip the following simulation states until the next conflict. Conflicts that have resulted in an alternative solution (as opposed to the previous one) will cause a different state to be created, and therefore the subsequent simulation states needs to be "calculated again". There is also a situation where alternative solutions have occurred within the conflict, but the original (external) solution is the same as in the previous simulation. Therefore, it is not true that the first change in the simulation would prevent further use of data from the previous level of simulation.



Simulation time
(Case C)
Conflict: 1
Variant: A
Conflict: 1
Variant: B
Conflict: 1, variant: A, replication: 0
Conflict: 1, variant: A, replication: 1
Conflict: 1, variant: A, replication: 2
Best variant (solution)
(Case D)
(Case E)
(Case F)
(Case G)
Simulation state
Solving of conflict situation (priority planning)
Solving of conflict situation (nested simulations)
Nested simulations variant
Equivalent states with previous simulation
New states (due to new level of recursion)
New states (due to changed solution of conflict)

Figure 6: Exact optimization algorithm progress

4.1. Algorithm

To implement the above procedure, it is necessary to change the original *simulation* function. It will now detect possible skips of the already solved states.

```
function simulation(s):
ps := loadPreviousSimulationStates(s)
while s.isNotEnd():
    if ps.isStateCompatible(s):
        state := ps.getStateForNextConflict()
        jumpToState(s, state)
    else:
        e := s.nextEvent()
        s.processEvent(e)

    if s.isConflictOccured():
        solution := solveConflict(s, s.getConflict())
        s.applySolution(solution)

    s.saveState()
```

4.2. Criteria for evaluating results

In order to determine whether this procedure is worthwhile, it is necessary to record the details of the nested simulation calculation progress - the consumed machine time and information about the individual decisions made in nested simulations.

4.2.1. Assessment of the decision-making process

As a first criterion for evaluating this process, it is possible to express the percentage of conflict situations that can be accelerated by this process. That is, to identify the last identical conflict state and to express the value of the ratio of identical conflicts between two calculations of nested simulations.

This criterion is very easy to calculate, but it does not provide a relevant metric for evaluating the actual impact on the required machine time for calculations.

4.2.2. Assesment of time savings

The time-saving criterion is based on the logic of the previous criterion. The criterion is not expressed by the relative number of conflicting states, but by the machine time needed to calculate simulation. Its value corresponds to the actual machine time that can be saved by remembering the previous level of simulations.

4.2.3. Assesment of memory requirements

Nested simulations represent a complex computational task whose computational demands grow exponentially (due to specific parameters). In order to make the simulation faster, it is necessary to remember the state of simulation in all conflicting states in all simulations (not only in the main simulation but also in all nested simulations). The criterion expresses the memory demand for one simulation, respectively the expected memory demands of all simulations. To effectively implement this method, it is necessary to have an external memory (with fast access) with the expected capacity (ideally operating memory, SSD, fast RAID array).

5. HEURISTIC METHOD FOR MERGING IDENTICAL REPLICATIONS

The *createReplications* algorithm method outlines how to create nested replications for each variant. For each variant, there are *ReplCount* simulations that differ by the default random number generator seed.

Heuristic for merging identical replications is based on the capability of prediction that the progress of two selected nested replications will be identical. If the match of two replications can be evaluated in advance, it may be decided to calculate only one replication.

Depending on the particular method of using this heuristic, it is possible to obtain a result whose quality is the same as the original procedure of nested simulations. This is possible when heuristics are used only at a particular level of nested simulations where no further recursion is allowed.

By using a heuristic at all nesting levels, the accuracy of the results can be reduced, and so decision-making process results in suboptimal solutions. In a situation where a conflicting state occurs in a nested simulation and at least one more level of recursion is allowed, using a heuristic may result in a suboptimal solution. Without a heuristic, each replication calculates the nested conflict using a new set of simulations (with different seeds of pseudo-random number generators). The use of a heuristic, limits the nested conflict solution to only one nested calculation and does not examine the complete original set of states.

Figure 7 shows an example of using that heuristic method. Here, the heuristic can be used to merge 2 replications for variant A_1 and 2 replications for variant B_1.

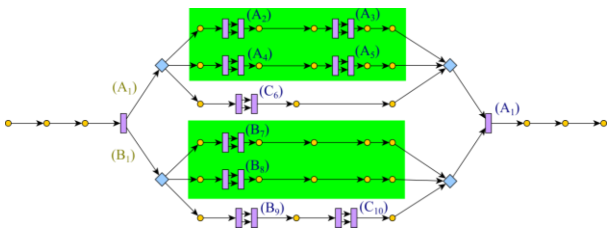


Figure 7: Example of possible replications merge

You can apply this method by modifying the original *createReplications* function:

```
function createReplications(variants, seeds):
simulations = {}
for variant in variants:
    for seed in seeds:
        s := createSimulation(baseSimulation, variant,
            seed)
        if (ss := existSimilarSimulation(s,
            simulations[variant]):
            increment weight simulations[variant][ss]
            delete s
        else:
            append s to simulations[variant]

return simulations
```

In a case study - a train station simulator - it is possible to detect the replication match in a relatively simple way. The nested simulation lookahead (run time) is defined by a constant. Two replications will be the same if the random events during the lookahead are the same. In this case, there is only one stochastic event "train arrival to simulation". It is enough to compare scheduled train arrivals between both replications to evaluate the match.

6. HEURISTIC METHOD FOR SOLVING SUBSEQUENT CONFLICTS

Typically, conflicts that occur during nested simulations (i.e. not in the main simulation), later may emerge in the main simulation (in general in a higher hierarchical level of simulation). This frequently leads to multiple calculations of the same conflict situations (see figure 8). A heuristic allows using results of these conflicts to immediately resolve future (identical) conflicts.

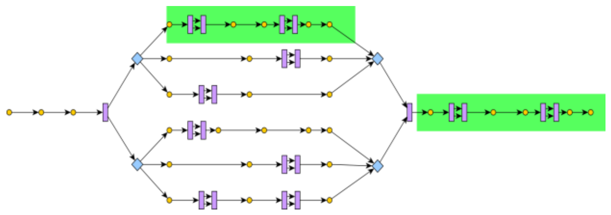


Figure 8: Example of subsequent conflicts

The heuristic can be applied by modifying the original *solveConflict* function:

```
function solveConflict(s, c):
if hasSatisfactoryPreviousSolution(s, c):
    return solveUsingPreviousSolution(s, c)

if simulation.recursionLevel <= REC_LIMIT:
    solution := solveUsingNestedSimulations(s, c)
    saveSolutionDetails(solution, s, c)
    return solution
else:
    return solveUsingAlternativeMethod(s, c)
```

The application of the heuristic is determined by a parameter which defines whether the previous result was calculated in sufficient quality and consequently it can be applied. As an example to select a criterion, the number of recursive levels of nested simulations can be specified. The previous conflict solution will be considered acceptable if the conflict has been solved with at least one nesting level. Thus, if the nested conflicts were allowed to be solved by nested simulations when dealing with a conflict, then previous conflict solution is usable. If nested conflicts were dealt with only by an alternative approach (priority planning), the new conflict would be solved in a standard way, and heuristics would not be applied.

7. TEST SCENARIO

For the purpose of conducting the initial case study (Diviš and Kavička 2017), the infrastructure of a smaller scale prototype railway station with several adjacent track sections (ending with simplified railway station models) was created. In particular, the operation of passenger services and a smaller range of freight transport are expected to take place at the station. In the case study, the behaviour of the station was tested at the arrival of delayed trains. For each train, alternative train paths were defined that use alternate station tracks.

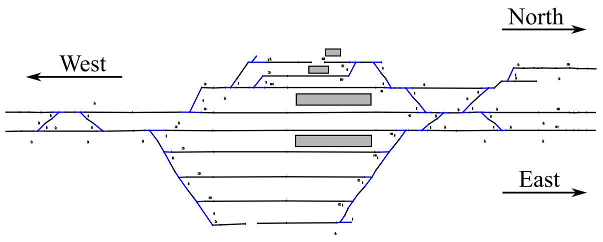


Figure 9: Schematic representation of central station and adjacent tracks

The representation of the main (central) station and the adjacent lines is shown in figure 9. There are two double-track lines (to the stations *West* and *East*) and one single-track line (to the station *North*) leading from the central station. The total distance between the eastern and western stations is about 20 km. The infrastructure is not completely fictional; it is inspired by several railway tracks and stations in the Czech Republic. The track profile contains sections with significant slope and arc ratios.
The simulation study focuses on the station traffic in a two-hour peak period. Passenger and freight trains are included, with the emphasis placed on passenger transport. Passenger transport is divided into two groups: (a) long distance transport - express trains; (b) regional transport - passenger trains. The traffic overview is shown in table 1. Figure 10 shows the occupation of station tracks in the central station under the conditions of deterministic simulation.
In the case study nested simulations are used as decision support. Its task is to select a replacement track for

Table 1: Parameterization of trains in the simulation model

Train type	Locomotive / vagons	Course	Interval between trains [h:mm:ss]	Total train count	Delay prob.	Delay mean time (exponential distribution) [s]
Express	1 / 7	West → Central → East	30:00	5	50 %	420
Express	1 / 7	East → Central → West	30:00	5	50 %	420
Passenger	2 / 4	West → Central → East	10:00	12	33 %	270
Passenger	2 / 4	East → Central → West	10:00	12	33 %	270
Passenger	1 / 2	West → Central → North	30:00	4	33 %	270
Passenger	1 / 2	North → Central → West	30:00	4	33 %	270
Cargo	1 / 22	West → East	1:00:00	2	50 %	1800
Cargo	1 / 22	East → West	1:00:00	2	50 %	1800

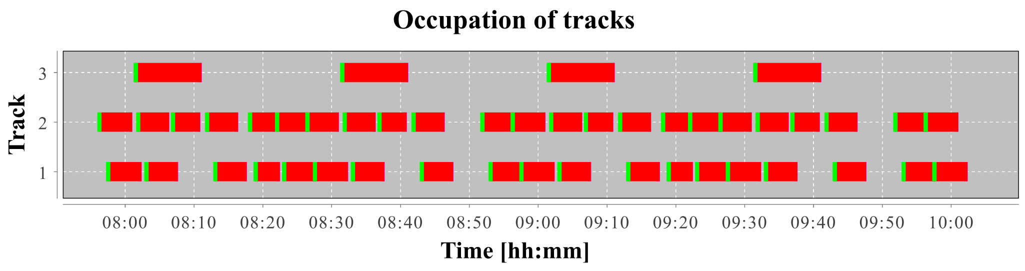


Figure 10: Occupation of station tracks during deterministic simulation

Delayed train, for which originally planned track is occupied.
In the case study, various parameterizations of nested simulations were tested and obtained results were compared. Basic parameterization of nested simulations:

- CrOpt – average weighted increment of train delays
  - weight is defined by a type (priority) of the train,
  - delay increment – the value is defined as the nonnegative portion of the train delay difference when leaving the simulation model minus train's input delay
- *CrOptComparer* – minimum function,
- *ScnCount* – without restriction,
- *ScnGen* – according to the available train paths,
- *ReplPreserveOriginal* – yes,
- *ReplResultsAggregateFunction* – average function,
- *RecStopCondBehaviour* – extension of the simulation time,
- *RecOnLimitAction* – usage of *priority planning* for conflict resolution,
- *FallbackScenarioConflictLimiter* – 15 s.

Varied parameters according to configuration:

- *StopCond* – 5, 15, 30 minutes,
- *ReplCount* – 3, 5 replications
- *RecLimit* – 0, 1, 2 levels of recursion.

For each test configuration was calculated 100 replications of main simulation.

8. SIMULATION RESULTS

Methods were evaluated based on detailed log files from full simulation runs. All methods were evaluated independently, one heuristic method at a time. Complex interactions between methods were not considered and evaluated.
Results are presented in ratio of simulations that are needed to process with heuristics with comparison to simulation without heuristics. Thus, results with lower values are better.

8.1. Exact optimization method

Because exact optimization algorithm is based on fact that you can skip parts of simulations from previous level of recursion, we evaluated effects on level 2 of recursion with information from level 1. From previous simulations, we found out that there is biggest increase in computational difficulty on level 2.
Unfortunately, exact optimization algorithm was evaluated as the worst method. Due to exponential growth of nested simulations, this method is able to save only up to 2 % of all necessary simulations. Method is also highly computationally expensive and requires nontrivial amount of memory. From these first trials this method couldn't be recommended to use.

Results are shown in picture 11. Detail of an analysis of a single replication is shown on picture 12.

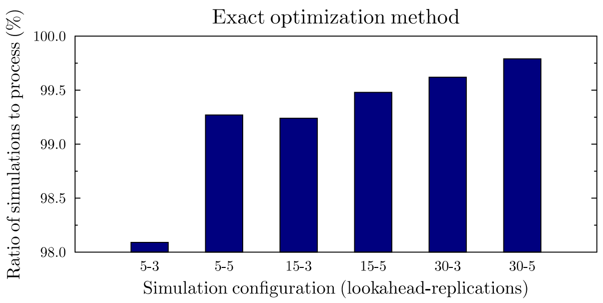


Figure 11: Results of exact optimization algorithm

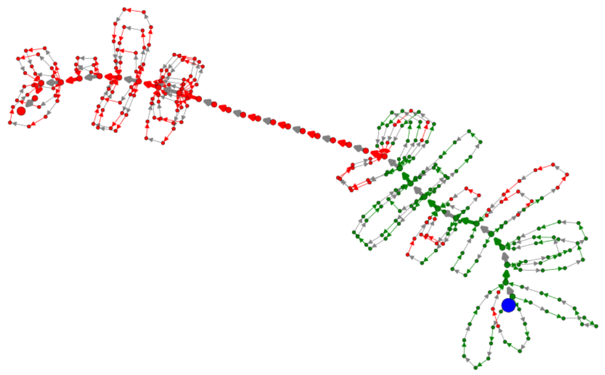


Figure 12: Detail of an analysis of a single replication – lookahead 15 mins, 3 replications, rep. no 38 – visualization shows state chart of the specific replication, blue circle represents starting state of a simulation, simulation of green states used exact algorithm optimization, red states must be recalculated due to state change in simulation

8.2. Heuristics - merging identical replications

Heuristics based on merging of identical replications appears to be very effective. With 5 minute lookahead it eliminated up to 60 % of all simulations, worst result of this method saved 25 % of simulations. With longer lookaheads effectivity of method decreased. With 30 minutes of lookahead it was able to eliminate up to 20 % of simulations.

Method was used and evaluated with 2 different approaches - (a) merging can be performed at all levels of recursion, (b) merging of replications can be performed only at leaf level of recursion. Method (b) should not negatively affect quality of results. Results are shown on picture 13 (method (b) is marked with 'L').

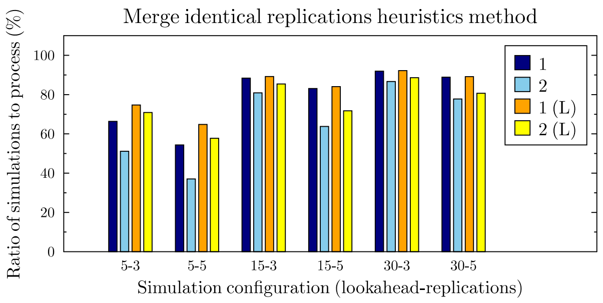


Figure 13: Results of merge identical replications heuristics with different levels of recursion allowed

8.3. Heuristics – solving subsequent conflicts

Heuristics based on solving of subsequent conflicts also appears to be highly effective. It constantly is able to eliminate about 40 % of simulations. Unfortunately, this heuristics method will result in suboptimal solutions. In evaluated scenario it appears that subsequent conflicts result to different solution in 20 % of times. Results of the method and ratio of matched solutions in subsequent conflicts is shown in picture 14.

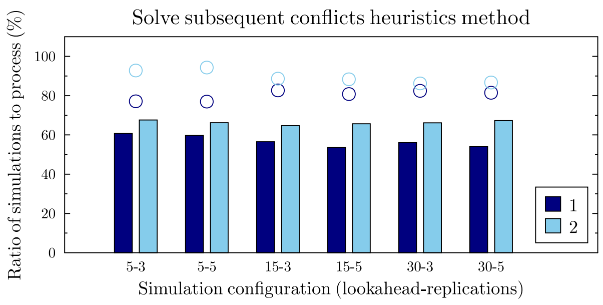


Figure 14: Results of solve subsequent conflicts heuristics with different levels of recursion allowed (bar charts shows actual results, circles show ratio of matched solution)

8.4. Comparison of methods

Results of rough comparison of all proposed methods are presented in picture 15. Numeric results of average value of necessary simulations are presented in table 2.

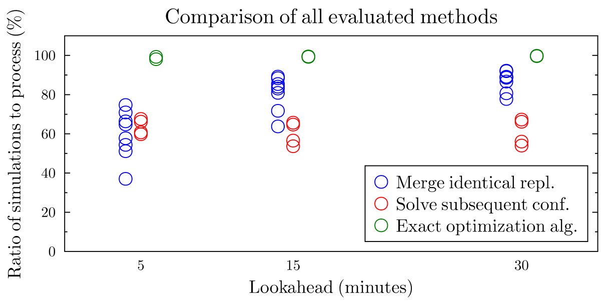


Figure 15: Comparison of all proposed methods

Table 2: Average number of simulations needed to process simulations with usage of specific method

Look-ahead	Repli-cations	Exact opt. alg.	Merge identical	Solve subsequent
5	3	98.1%	65.8%	64.2%
	5	99.3%	53.5%	63.0%
15	3	99.2%	86.0%	60.6%
	5	99.5%	75.7%	59.7%
30	3	99.6%	89.9%	61.1%
	5	99.8%	84.1%	60.7%

9. CONCLUSION

Results shown big differences between proposed methods. The exact optimization algorithm appears to be highly ineffective due to exponential growth of new simulations needed to process in new level of recursion. High complexity of the algorithm, overhead, low effectivity and memory demands rule out this method from real world usage.

Other two heuristics methods appear to be quite usable and results in saving up to 40 % of necessary simulations. Unfortunately, quality of results may decrease due to application of these methods. There are few possibilities how to apply these methods to maintain same quality of results. But extended study of interactions between these methods and how results will be affected, remains as a task to further studies.

ACKNOWLEDGMENTS

The work was supported from ERDF/ESF *Cooperation in Applied Research between the Univ. of Pardubice and companies, in the Field of Positioning, Detection and Simulation Technology for Transport Systems - PosiTrans* (CZ.02.1.01/0.0/0.0/17_049/0008394).

REFERENCES

Bonté B., Duboz R., Quesnel G., Muller J. P., 2009. Recursive simulation and experimental frame for multiscale simulation. In: Proceedings of the 2009 Summer Computer Simulation Conference, 164-172. July 13-16, Istanbul, Turkey.

Diviš R., Kavička A, 2015. Design and development of a mesoscopic simulator specialized in investigating capacities of railway nodes. Proceedings of the European Modeling and Simulation Symposium, 52-57. September 21-23, Bergeggi, Italy.

Diviš R., Kavička A, 2016. The method of nested simulations supporting decision-making process within a mesoscopic railway simulator. Proceedings of the European Modeling and Simulation Symposium, 100-106. September 26-28, Larnaca, Cyprus.

Diviš R., Kavička A, 2018. Complex nested simulations within simulators reflecting railway traffic. Proceedings of the European Modeling and Simulation Symposium, 178-186. September 17-19, Budapest, Hungary.

Gilmer J. B., Sullivan F. J., 1999. Multitrajectory simulation performance for varying scenario sizes [combat simulation]. In: WSC'99. 1999 Winter Simulation Conference Proceedings. 'Simulation - A Bridge to the Future' (Cat. No.99CH37038), 1137-1146. December 5-8, Phoenix, AZ, USA.

Gordy M. B., Juneja S., 2010. Nested Simulation in Protfolio Risk Measurement. Management Science 56:1833-1848.

Hill J. M. D., Surdu J. R., Ragsdale D. J., Schafer J. H., 2000. Anticipatory planning in information operations. In: SMC 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions' (Cat. No.00CH37166), 2350-2355. October 8-11, Nashville, Tennessee, USA.

Kindler E., 2010. Nested Models Implemented in Nested Theories. In: Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling & Simulation, 150-159. May 29-31, Catania, Sicily, Italy.