ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20190723085606
Preliminary communication

Simulating Train Dispatching Logic with High-Level Petri Nets

Dušan JEREMIĆ*, Sanjin MILINKOVIĆ, Sandra KASALICA

Abstract: Railway simulation is commonly used as a tool for planning and analysis of railway traffic in operational, tactical and strategical level. During the simulation, a typical problem is a deadlock, i.e. a specific composition of trains on a simulated section positioned in such a way that they are blocking each other's paths. Deadlock avoidance is very important in the simulation of railways because deadlock can stop the simulation, and significantly affect the simulation results. Simulation of train movements on a single track line requires implantation of additional rules and principles of train spacing and movement as train paths are more often in conflict than on a double track line. A High-level Petri Nets simulation model that detects and manages train path conflicts on a single track railway line is presented. Module for train management is connected to other modules on a hierarchical High-level Petri net. The model was tested on a busy single track mainline between Hrpelje-Kozina and Koper in south-western Slovenia.

Keywords: deadlock prevention; high-level Petri nets; railway simulation; train dispatching logic

1 INTRODUCTION

Railway traffic is considered a very complex system as it consists of interactions between trains, infrastructure, and constraints imposed by transport demand (timetable). Trains operate under the operational procedures, rules (dependent on the type of safety equipment installed on the track) and interactions with other trains. The interaction between trains is planned in a timetable, a schedule of train movement. The planned timetable is determining the departure and arrival times for all trains, but this plan is always influenced by real-life stochastic disruptions, so often trains operate with delays. Movement of trains on a single track line is additionally restricted as, at one point in time, only trains moving in one direction can be occupying a track section between two stations. The conditions in these complex systems are very hard to analyse without the use of simulation modelling. Simulation of railway operations and train movement can be a demanding task and depends on the level of details that are required to analyse. In the microscopic simulation, a model of a railway line, station or a network, is replicated with a high level of details, including, for example, track grade, curvature, tunnels, location of power supply stations, etc. For a macroscopic level, selection of elements, combination of objects, or simplifications and assumptions, can be selected depending on the level and size of the simulated system. Many software tools and simulation packages are developed for this purpose and offer a specialized approach to railway simulation.

Simulation is a very efficient tool for the analysis of the train traffic and thus an important input to a process of planning in railways. Railway simulation is mostly used to estimate timetables, the capacity of lines, train delays, train stops, energy consumption, etc. When planning a new line or building new railroads, it is an essential tool for analysing the impact of the new system. Petri nets are used in many applications that have procedures for dynamic discrete event systems, such as manufacturing, industry, and transport systems. Petri net can simulate systems with reasoning about objects and resources and their changing states. Petri nets are well suited to represent the states of a dynamic system made up of concurrent active objects sharing resources. Consequently, Petri nets are selected as a tool that can be used because of its ability to model dynamic discrete event system, and the system of train movements on a single track line is such a system.

This paper addresses the problem of deadlock avoidance for railway simulation on a single track line. This is important as railway simulation is commonly an integral part of planning processes in the operational, tactical and strategic level. Module for dispatching logic is connected to subsystems of all stations in a model. Deadlock avoidance during simulation of train's movement is managed in a dispatching logic module. Dispatching logic is implemented by using Petri nets ability to model the concurrent behaviour of distributed systems. In this paper, a simulation model of a single track rail line based on High-level Petri nets (HLPN) is proposed. Further, the condition of a train movement between stations on a single track is in Petri nets presented by its ability to coordinate concurrent active objects that share resources. Model is created by modules defined for a specific type of sections, connected in a hierarchical structure. The presented model uses High-level Petri nets where trains are tokens, places are sections (track section, station section, switch section, etc.), transitions are conditions for train movement. Properties of HLPN are used to organize and simplify the model: colour tokens have information on train number, path, length, category, etc.; hierarchy enables the design of modules that represent sections and combination of a section in different levels of modelling, time is used in managing the departure of trains according to the timetable. HLPN model is further developed to model dispatching logical reasoning during simulation by collecting data on train location. Data on track section occupation, required to create dispatching decisions, are collected from the station's subsystems by tokens that indicate the place status. Module for train dispatching can alter train paths and give priority for specific train depending on the current traffic situation: number of available tracks in neighbouring stations, location of the trains, and train priority. Verification and validation of an HLPN model are achieved by comparison to the simulation model created in OpenTrack commercial software, for a selected single-track rail line between Hrpelje-Kozina and Koper in south-western Slovenia. This line, although simple with only three stations and one junction was

chosen due to the fact that it is used by a high number of trains, over 100 trains per day are scheduled. On the other hand, stations have low capacity, just one passing track per station and using their capacity to the maximum is necessary for the functioning of the line.

Paper is organized as follows: after the introduction and literature review, in Chapter 3, the High-level Petri net model of single track train traffic is presented. In Chapter 4 module for simulation of train dispatching logic by High-level Petri Nets is presented. In Chapter 5 a case study: a simulation model of a Hrpelje-Kozina-Koper line is presented. Then, in Chapter 6 simulation results and discussion, and finally, in the last chapter, conclusion is presented.

2 LITERATURE REVIEW

There is a number of software packages that are specialized for railway operation simulations: OpenTrack, RailSys, RTC, MultiRail, CMS, etc. These software solutions are very good for general simulation, very easy to use, and give excellent results. However, they cannot be easily adaptable for specific problems and complex systems without the use of simplifications, assumptions, and reduction of a model for railway systems. As a simulation tool, Petri nets are used in many applications including industrial, manufacturing and other transportation systems which share common principles with railway traffic. All these systems share the common goal of avoiding deadlocks or states in which the system could not operate beyond a certain point.

Petri net theory and applications are discussed in several papers by Jensen and Rosenberg [1], Jensen [2] and He and Murata [3]. Yang and Li [4] developed dynamic timed fuzzy Petri net used for monitoring and control of industrial processes. Janssens et al. [5] used a Timed Petri net for vehicle routing simulation. Xing et al. [6] developed a Petri net controller for preventing deadlocks in flexible assembly systems. Yue et al. [7] analysed automated manufacturing systems using Petri net model in case of unreliable resources and failures. Liu et al. [8] used stochastic Petri net of subsea blowout preventer control system.

In railway applications, Petri nets are often used for railway infrastructure and timetable models. Yung-Hsiang Cheng and Li-An Yang [9] developed a decision support system for train dispatchers using fuzzy Petri Net. Fuzzy Petri Net approach is used to formulate decision rules of dispatchers in case of abnormality in which decision rules and factors are collected by interviews with dispatchers. Meng et al. [10] used Petri nets to create macroscopic network model of train operations in the railway network. The model consists of train tracks, station, and arrival-departure track subnet model. Model is used for harmonizing train timetables of neighbour dispatching sections. Malavasi and Ricci [11] created modular Petri net model with generalized station model for any layout or interlocking system and generalized line model to be applied to any signalling system. Station model contains sets of nodes which represent various elements of railway infrastructure: track circuits, switches, signals which are linked with transitions. Line model contains sets of nodes such as block sections, signals which are also linked with transitions. Potehkin et al. [12] developed a discrete event model using Petri nets. The model consisted of several basic elements such as station to station block, segment, block section, railway point and a light signal. Special control elements ensure railway traffic safety requirements. Fay [13] developed a fuzzy dispatching support system, which analyses traffic for near future and proposes adequate dispatching actions based on expert knowledge. Knowledge is acquired with interviews and open questions with dispatchers, which is used to define rules represented with Fuzzy Petri Net. Zhuang et al. [14] modelled train timetable with timed place Petri net and used it for conflict prediction. Conflict prediction is shown by means of train graph with suggestions to dispatchers how to resolve the conflict.

Deadlock detection, analysis, and prevention as a crucial part of any railway infrastructure simulation are analysed in several papers. Fanti et al. [15] used a coloured Petri Nets (CPN) model of a dynamic rail system for determining deadlock situations. The prevention policy is expressed by a set of linear inequality constraints, called coloured Generalized Mutual Exclusion Constraints that are enforced by adding appropriate monitor places. Using digraph tools, deadlock situations are characterized and a strategy is established to define off-line a set of Generalized Mutual Exclusion Constraints that prevent deadlock. Pachl [16] proposed a method for deadlock detection and avoidance with dynamic route reservation. In order to authorize the train entrance to the next section, a specified number of track sections in front of a train must be reserved. Track sections could be reserved for several trains in which case reservations are stacked in a similar way like stacked routes in an interlocking.

The model presented in this paper is an extension of a model described by Milinkovic et al. [17], where authors developed a Fuzzy Petri net (FPN) for estimating train delays. Based on similar modelling principles, the use of a simulation model is extended by introducing the HLPN subsystem that uses logical reasoning of a train dispatcher to create a deadlock-free simulation on a single track railway line.

3 MODELLING TRAIN MOVEMENT BY HIGH-LEVEL PETRI NETS

Petri nets are a graphical and mathematical modelling tool used to analyse and simulate concurrent systems. Theory of Petri nets is based on the mathematical theory of bipartite graphs. The system is modelled as a bipartite graph with two sets of nodes, a set of places which represent state or system objects and a set of events which represent transitions which determine dynamics of the system [18].

3.1 High-Level Petri Nets

High-level Petri nets (HLPN) is a term which is used for many extensions of basic Petri net principle such as coloured Petri nets, hierarchical Petri nets and others. HLPN have much higher modelling power since all tokens can be assigned with its own colour which has a certain value of a complex data type. Token colours enable the use of complex logic expressions and functions which can be

used on places, transitions and arcs. In addition, the hierarchical structure of HLPN enables the use of subsystems or modules by which large models can be obtained by combining a set of subsystems. Definition of High-level Petri net used in this paper is given in Milinkovic et al. [17].

A Petri net is a 6-tuple $PN = (P, T, A, W, M, D)$, where:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions;
- A is a set of directed arcs connecting places and transitions;
- W: $A \rightarrow \{1, 2, 3, \ldots\}$is a weight function for labelling arcs, where a k-weighted arc denotes the set of k parallel arcs;
- M: $P \rightarrow \{0, 1, 2, 3, \ldots\}$is the initial marking assigning to place p a nonnegative integer k, i.e., marking place p with k tokens;
- D: $P(DS) \rightarrow \{0, 1, 2, 3, \ldots\}$is the initial marking for PN dispatching system assigning to place $p(ds)$ a nonnegative integer $k(ds)$, i.e., marking place $p(ds)$ with $k(ds)$ tokens.

On PN graphs places are represented with circles while transitions are represented with squares or rectangles. The arcs for input and output functions are represented as directed lines. The transition of a system from one state to another occurs when an event transpires, it can also be a moment when the time period in a certain state expires.

3.2 Modelling Railway Operations Using Petri Nets

Simulating a railway infrastructure is often a complex and demanding task in which many dependencies and variables must be taken into account in order to create an accurate model. Depending on the desired accuracy infrastructure can be modelled as microscopic or macroscopic level. The macroscopic level is more general, stations are represented with single nodes and described with attributes regarding the station capacity such as a number of tracks, speed and other relevant information. Microscopic model is more detailed since all basic elements such as tracks, switches and signals are represented with separate nodes which are linked together with dependencies as in the real system.

In Petri nets trains can be represented as tokens while all other infrastructure elements (block sections, switches, station tracks) can be represented as places. Since tokens are coloured, they can carry a number of information about train attributes which are constant trough model like train number, length, train type but also some information which can be changed as the train (token) passes through the model. That information includes arrival, departure and waiting times at each place, arrival and departure speed at each section, train path and other relevant information concerning train movement through the system. Besides trains, some other elements can be represented as tokens, for example, information carriers between different elements of railway infrastructure. These tokens can also be coloured depending on types and amount of information which they carry. The information which includes time can only be carried on timed Petri nets.

Transitions in Petri nets are used for applying firing rules or rules for movement of tokens through the model. There is a number of rules with different level of complexity which can be used. Simpler rules usually do not include token colour or other properties, for example forbidding firing in case output place is busy. These rules are often embedded in simulation software and do not need additional defining. Advanced rules use token colour as a decision criterion, for example, train number, length, direction are used to determine if the transition would occur or not. In case of multiple output places these rules decide which transition should occur.

There are two conditions which Petri net needs to fulfil in order to accurately simulate railway infrastructure and that is safety and liveness condition. Safety conditions are basic conditions in railway traffic, for example, two moving trains cannot occupy the same track section at the same time or train routes could not overlap or traverse each other. Liveness condition means that in no circumstances deadlock between trains could occur. Possibility of deadlock is much higher on single track than on double track lines since on single track lines there is only one track for both directions of travel.

3.3 Modelling Elements of Railway Infrastructure

Model of railway infrastructure for this research was created with software ExSpect, which is used for discrete process modelling. It is best-suited for modelling of production and logistics chains, business processes and telecommunication systems. ExSpect supports hierarchical and coloured Petri nets which enable the use of subsystems in the modelling process. Models can be analysed for structural correctness properties, and their behaviour can be observed through simulation, either step-by-step, continuous or with breakpoints.

Program components include predefine elements which can be used to create objects in the model. Basic elements are places (channels), transitions (processors) and arcs (connections) which are standard Petri net elements. There are also elements like the store which are used to save and export data during simulations, time generator and input and output pins, which are used to connect lower to higher level subsystems inside a model.

In this paper the ability of HLPN to create hierarchical systems is used. The whole model is decomposed into subsystems or modules which are stored in the program database and can be easily imported into the model. This is a very useful feature for models where some elements or subsystems appear more than once. In railway infrastructure simulations those elements are a block or track section and switch. Track section corresponds to insulated section, which is section of a track whose occupancy can be controlled independently from other sections. Model is then created by entering modules in order as in real system and then connecting them to each other and to other necessary elements such as stores. Such a method requires more time during initial programming for module creation but enables the creation of larger and more complex models since these modules can be used to model any system where there are similar processes. By creating additional modules it is possible to model a large number of railway systems.

3.4 Modules in Petri Net Simulation

Basic modules which are used in modelling any railway line are block section and switch. The main difference between them is that the block section has one input and one output for each direction and switch has a minimum of two outputs for at least one direction of travel. A most common case for the switch is two outputs in one direction of travel which represents a simple one-sided switch in a real system. Block sections can be either line block sections between stations or track sections in stations, where common elements include input and output for each direction, elements which calculate travel time through section and section occupancy. In order to make this module universal for different block sections, all block section modules need to have external connections for data different for each block section like length and permitted speed. Other external connections which appear in both kinds of block section include time generator and storage "state" which is used for keeping information about the state of the block section (free, occupied by train route, occupied by train). The main difference between the line block section and track section module is in speed, travel time calculation and block release elements. Line block section calculates these elements from block signal aspect while track section receives them from external storage. Simple switch module is shown in Fig. 1.

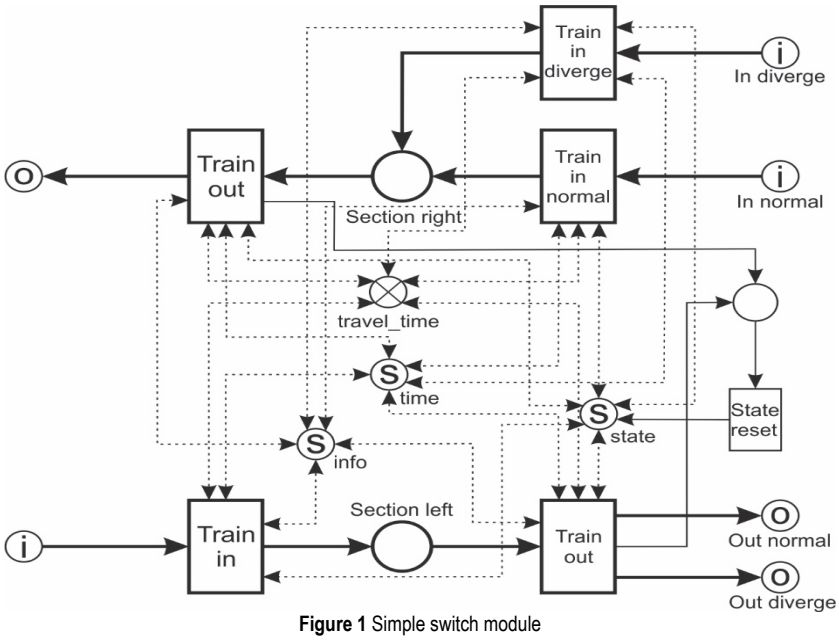

Figure 1 Simple switch module

Model of railway infrastructure is created by placing these two types of modules in order in which they appear in the real system. Modules are connected directly to each other or via transitions that represent main signals. These modules also need to be connected with their other external elements such as time, state, etc. Since these modules are just executory devices, they need control logic in order to successfully manage train movement. This control logic is contained in the dispatching module as main control logic and in station interlocking processors that execute commands from the dispatching system.

3.5 Simulating Railway Traffic by High-Level Petri Nets

In railway operation, there are several basic rules which are mutual for all railway systems. The first and most basic rule is about track occupancy: two moving trains cannot occupy the same track section at the same time. Since for all train movements precise routes must be set, there are also several rules which govern relations between routes, such that train routes could not overlap or traverse each other. These are basic rules which could be easily applied in Petri nets with some IF-THEN dependencies and they can be performed by station interlocking processors.

Station interlocking processors are transitions that receive route setting commands and send tokens to other infrastructure elements to set appropriate state. Route setting commands are actually multi-coloured tokens with all information necessary for route setting such as: the direction of travel, track number, type of route, train number of a train which requests route setting and state of the route setting command. Route setting command starts when the train passes certain signal, it is then sent to the dispatching system, and then to station interlocking processors. When route setting command (token) is received, the first step is to check the state of all elements of the desired route and if they are free setting their state to "occupied by route". If either element is busy route setting command is returned to the dispatching system with the state "busy".

4 TRAIN DISPATCHING LOGIC BY HIGH-LEVEL PETRI NETS

Station interlocking processors are suitable just for basic tasks, executing commands received from dispatching system. On single track lines with bidirectional travel direction, the more complex control logic is required and therefore separate dispatching subsystem must be used. In this model dispatching system is created as a

separate module which is connected to the rest of the system by input and output pins used for sending and receiving command tokens. Dispatching system itself has several components which are divided by type and status of incoming command. Type of command represents the type of route and in most stations the only entrance and exit routes are used. The state is another characteristic and all commands could have several possible states:

1) Request-request for route creation
2) Ok-the route is successfully created
3) Busy-route or part of the route is occupied
4) Free-the route is released and freed for another use

All commands enter the dispatching system through a single processor ("Command selector") and are routed to the appropriate subsystem depending on their type and status. In all subsystems, the first level of control is checking whether requested route setting command is available and not in conflict with another command (route). Deadlock prevention often requires interaction between two subsystems. Overview of dispatching subsystem is shown in Fig. 2. For simplicity and better understanding, some subsystems are shown as one transition (square) but actually, they include several transitions (processors) that perform different checks.

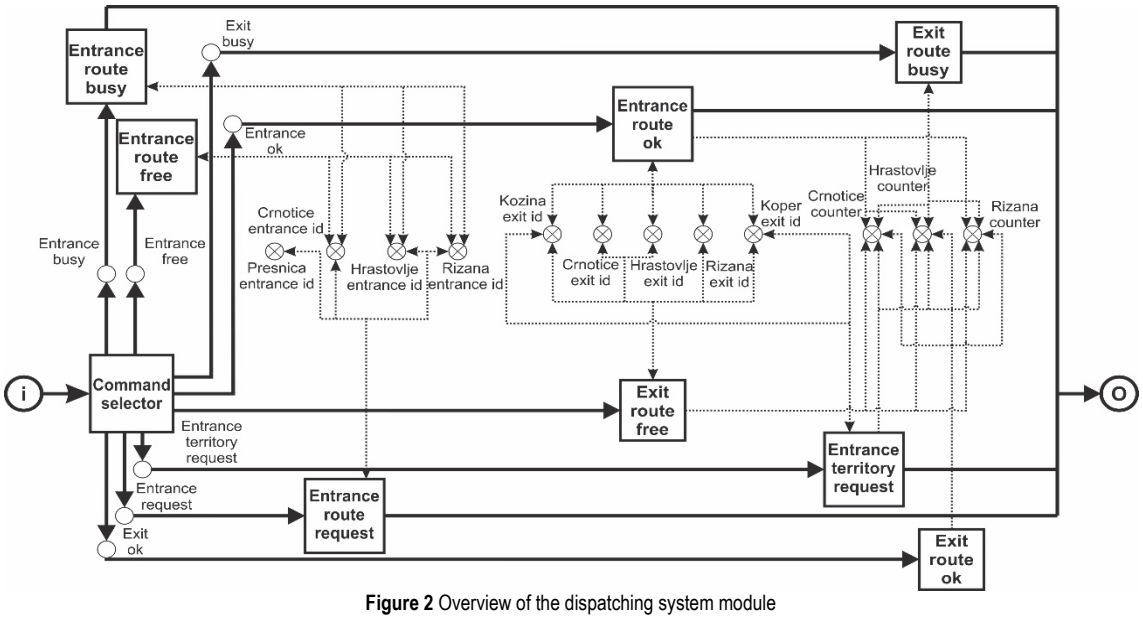

Figure 2 Overview of the dispatching system module

Subsystem "Request" handles commands for entrance route requests in stations and performs several checks before sending route request to station interlocking processors. The first check is the availability of requested command in "entrance id" storages; if the requested route is not available then that route has already been used by another train. Next check is performed to establish if selected route could cause deadlock by checking number of available track in the stations and whether the route to the same track from opposite direction is used. And final check is determining if there is a train from opposite direction, in that case route is changed to siding instead to main track. Similar in function is "Entrance territory request" subsystem which performs check whether the entrance of a train to simulated section is permitted.

Subsystem "Ok" handles commands for entrance and exit routes after they are successfully formed. In case of entrance route, this command triggers a request for an exit route from the same track in the same direction. That command is then redirected to exit command subsystem.

Subsystem "Busy" handles commands of routes which could not be successfully formed due to the occupation of some part of the requested route. In case of entrance route in most cases occupation is caused by route overlap or flank protection from another route and in case of exit route usually next block section is occupied.

Subsystem "Free" handles commands from released routes. Route release for both entrance and exit routes occurs when the train passes the exit signal. This subsystem returns route id to appropriate storage and therefore enables usage of that station track for another route.

Deadlock occurs when two trains from opposing direction meet and neither of them could continue its journey. This problem is more pronounced on single track lines since there is only one track for both directions of travel. The deadlock could occur on the open line between stations or in the station area. Deadlock on open line could happen when the line is divided into two or more block sections, in which case trains could meet on an open line. This problem is easily solved with travel direction blockade. This solution is part of every modern interlocking device and in this model is also incorporated in the station and block interlocking processors.

A more serious and complex problem is when a deadlock occurs in the station area, a problem caused by a number of trains exceeding the station's capacity. A common case of deadlock is when all tracks in the station are occupied with trains going in the same direction, and at the same time, train from the opposite direction is travelling towards that station. Basic principle in deadlock prevention is that number of opposing trains from at least one direction must not exceed number of free tracks in the station minus one track for opposing trains. Principles of deadlock detection and resolution are shown in Fig. 3.

In this model, the proposed deadlock detection system is somewhat similar to the system proposed in Pachl [16], but instead of route reservations it uses capacity counters for each station and by comparing their state determines if

deadlock could happen. Capacity counters are storages which keep numerical information about the number of trains in the station, number of trains approaching the station and number of trains which have exit route formed from next station, for both directions of travel. In dispatching system, these numbers are connected with route forming and releasing. When the exit route is formed from one station, capacity counter in the next station for that direction of travel is increased by one. When exit route is released from one station, capacity counter for the same station is decreased by one. Deadlock prevention is therefore based on comparing states of the next station counter and determining if the next station has enough capacity to accept train. If not, the train is held at the current station and exit route request is repeated at regular intervals. Beside stations, entry points to a simulated section also need to check for possible deadlock before accepting trains. When a train is entering the simulated territory, dispatching system must check every station capacity counter in order to determine if there is enough capacity to accept another train. Checking starts from the first station and if it has not enough free tracks checking is performed on next one until the first station with enough free tracks is found. Checking is performed using IF – THEN dependencies and if the command for exit route could create deadlock command is delayed until there is no more possibility for deadlock. The same principle applies for route request for entering a simulated route. By using capacity counters entrance routes to stations are not formed far in advance but only when the train is at a certain distance in front of the station, in this paper it is two block sections in front of the station.

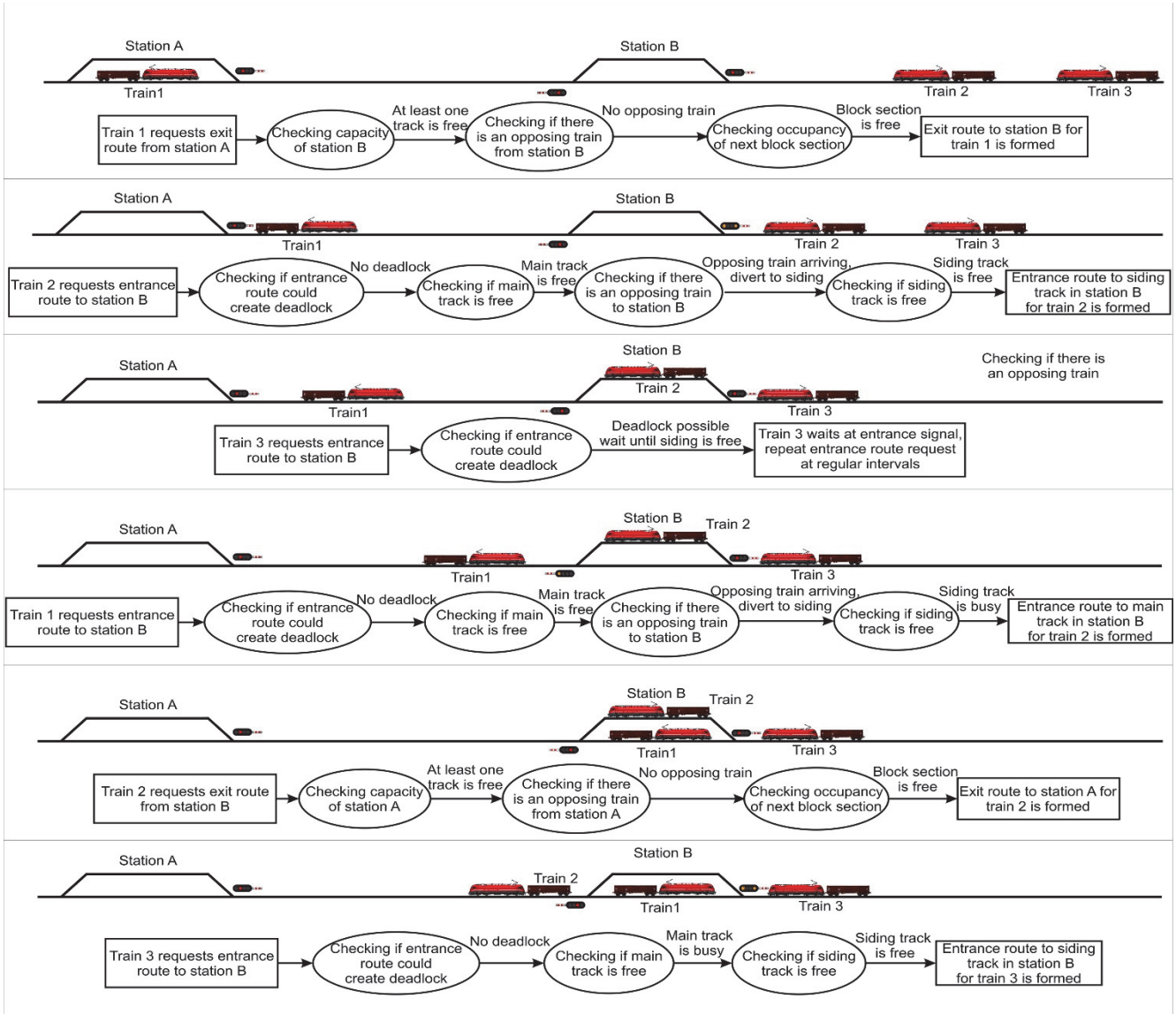

Figure 3 Illustration of deadlock system functioning

5 SIMULATION OF RAILWAY LINE HRPELJE-KOZINA-KOPER

Railway line Hrpelje-Kozina-Koper is located in south-western Slovenia, it is a part of Divača-Koper line which connects Ljubljana-Sežana international main line to the port of Koper. Nowadays Koper is one of the busiest Adriatic ports and it is used by trains from Slovenia, Austria, Germany, Czech Republic, Slovakia and Hungary which in turn makes Divača-Koper line the busiest railway line in Slovenia. Due to difficult terrain and height difference the line is still single tracked with crossing sidings and a maximum gradient of 26‰. These are the main reasons why this line, with around 100 trains per day,

has capacity utilization over 90% of its theoretical capacity.

Currently, on this line, there is one junction Prešnica and three stations: Črnotiče, Hrastovlje and Rižana. All three stations have two tracks used only for meeting and overtaking of trains. The line between stations is divided into block sections with three block sections between each station, the only difference is between Črnotiče and Hrastovlje where there are four block sections. Traffic is controlled from dispatching centre Postojna which is located on Ljubljana-Sežana mainline. Schematic of the line with all tracks, switches and signals is shown in Fig. 4.

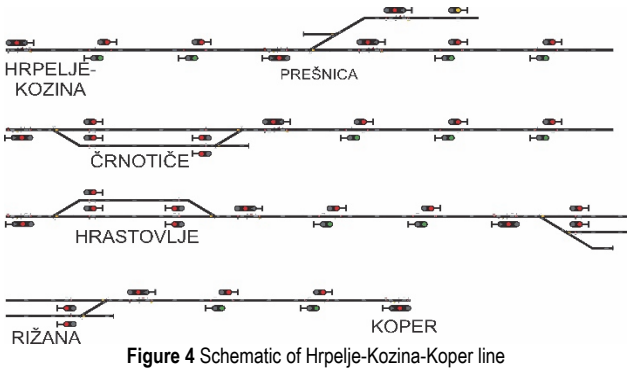

Figure 4 Schematic of Hrpelje-Kozina-Koper line

5.1 Petri Net Model of Hrpelje-Kozina-Koper Line

The basic principle in constructing Petri net model of this line was using modules described in previous sections, connecting them in the order in which they appear in a real system and connecting all modules with the station and block interlocking processors. Track sections in the station are also connected to additional processors, such as track release processors. After connecting modules to each other, additional elements necessary for their functioning are added. Those elements include storages info and state which are required for all modules but also storages sig, speed and direction which are required for certain modules. All these storages require predefined initial value. Processors which represent signals are placed at the end of each section, block signals are placed between block section modules, entrance signals in front of the station and exit signals at the end of each track section in stations. Their purpose is to allow or forbid the passage of trains (tokens) and to return that signal to default aspect after the train passes. All stations have a separate interlocking processor which receives route requests from dispatching subsystem.

When all elements are connected the last step before running the simulation is adding trains. Trains can be added manually by entering data to first place at each end of the model or by importing data about trains from the .txt file. Train data needs to be written in a .txt file in a specific format and sequence of data.

In order to verify this model, the same section was modelled using commercial software OpenTrack, a microscopic synchronous railway simulation model. OpenTrack simulates the behaviour of all railway elements (infrastructure network, rolling stock, and timetable), as well as all the processes between them. It can be easily used for many different types of projects, including testing the stability of a new timetable, evaluating the benefits of different long-term infrastructure improvement programs, and analysing the impacts of different rolling stock. For verification and comparison of results obtained by both simulation models OpenTrack model of Hrpelje-Kozina-Koper railway line is used.

5.2 Simulation Parameters

Railway line Divača-Koper is the busiest line in Slovenia so it was possible to use real data to fully test the model. For testing of this model the timetable from year 2016/2017 was used with approximately 100 trains per day which is close to the line theoretical capacity. For more extensive testing, variants with 150 and 300 trains per day and with random train arrivals were used.

ExSpect has a possibility to export data into an Excel file and in this model data from each channel between sections are entered into single Excel file. Information from places which is entered into single .xls file is mostly used to review the functioning of the whole system, delays, most used routes, travel times and speed. Every entry includes data about a train that usually do not change while the train is passing through a model like train length, train type, destination and data which changes at each place that train passes like current train path and current position. All records include simulation time at which they were recorded. All data about train movements and track occupancy saved in excel can later be exported to AutoCAD for graphical representation.

During the simulation, progress could be observed through custom animation which is created by connecting elements from model to bitmap images. Elements like switch, block section and signals are represented with different bitmap images and every change in their state is shown as a change of the bitmap image.

5.3 Simulation Results and Discussion

The goal of this simulation was to create an accurate model of busy single track railway line using Petri nets and to develop a dispatching system which could manage railway traffic without creating deadlocks and unnecessary delays. Accuracy of the model and efficiency of the dispatching system was verified with a model of the same section created in OpenTrack. Simulation results were shown in the form of train graph shown in Fig. 5. Red lines represent planned timetable, blue lines simulation results from Petri nets and green lines represent simulation results from OpenTrack.

The model was able to simulate conditions with 150 and 300 trains per day without deadlocks, but those results were not compared with OpenTrack or planned timetable since models with 150 and 300 trains used random train arrivals and were used only for testing purposes. First comparison and verification were performed in situations where there are no meeting of trains. As an example the time period from 0:00 to 3:30 was used. The second comparison was performed in cases when opposing trains meet. In both cases, the solution of conflicts is different due to different arrival times in stations. Different arrival times are on one hand created by different travel times and on the other by interactions with other trains in previous stations.

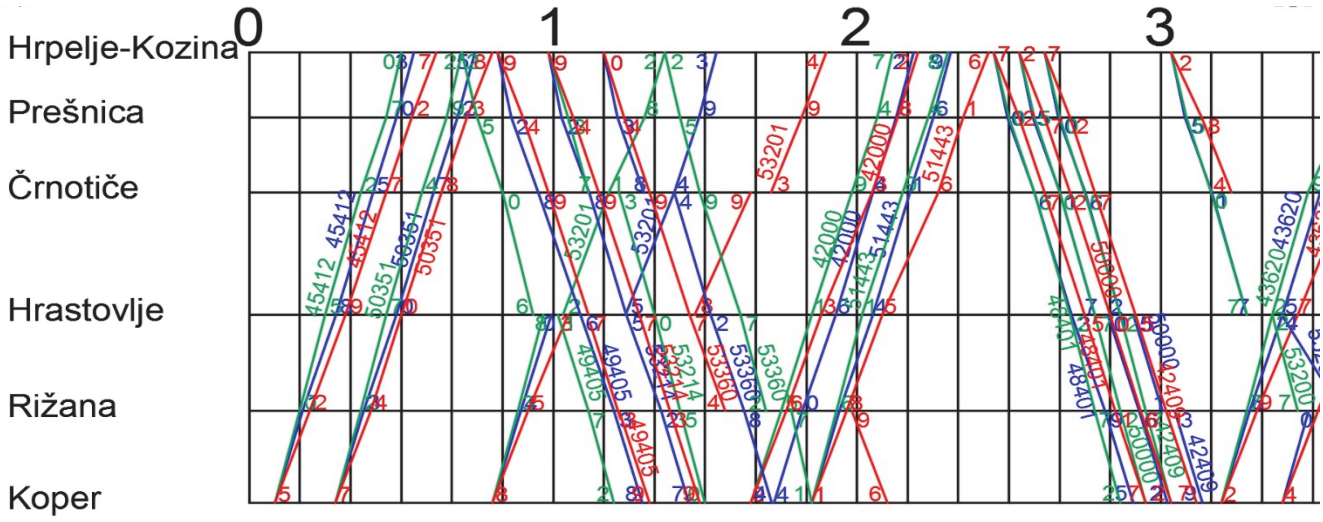

Figure 5 Part of the train graph of Hrpelje-Kozina-Koper railway line from 0:00 - 3:30

In all cases, it can be observed that stop time in both models is lower than in the train schedule. For example, train 53360 in train schedule waits 25 minutes, in OpenTrack simulation waits 5 minutes and in Petri net simulation does not wait at all. Results show that the duration of stops and delays in stations is kept at a minimum in both simulation models, and in many cases is shorter than a train schedule. The third comparison was performed by the length of stopping times at all stations. Stop durations are shown for Petri net model, OpenTrack model and train schedule. For all three models total stop time, average stop time and maximum stop time are shown in Tab. 1.

Table 1 Duration of stop time per model

Model	Total stop time / h:mm:ss	Maximum stop time / h:mm:ss	Average stop time / h:mm:ss
Petri net model	3:24:00	0:16:45	0:06:11
OpenTrack model	3:33:00	0:18:00	0:06:27
Train schedule	17:56:00	1:04:00	0:17:56

Results show significant improvement in both models over stop time obtained from train schedule, in total stop time as well as in average and maximum stop time. When comparing the two models, the Petri net model shows shorter total stop time as well as maximum and average stop times. These results can be confirmed by analysing train graph which shows the different solution of conflicts but the similar duration of stops.

Performance of dispatching subsystem itself can be evaluated by comparing a number of dispatching actions in case of different train arrivals. For this task current train schedule with 100 trains per day was used, but with random delays for all trains. In order to obtain more accurate and average per day results 30-day period was used with different delays for all trains. The model was tested with a different range of random delays, which were generated in Excel with a random number function. Three dispatching actions were observed: train stopped on the main track, train diverted to siding, and train diverted to siding and stopped. The train stopped on main track was used in a situation when the exit route could not be formed due to insufficient buffer times between consecutive trains or in case the trains meet. Diverted in the station is an operation used in the case that main track is occupied by another train when exit route for the diverted train could be formed. Stopped and diverted in the station is sometimes used in cases mentioned above but mostly for meeting of an opposing train. Since the default route setting was through the main track, every time train is diverted or stopped means that the dispatching system performed an action other than just sending received command to interlocking processors. Number of dispatching actions in 30-day period per station is shown in Tab. 2.

Results show that in most cases the number of dispatching actions increases with the increase of random delay intervals. In case of complete random arrivals number of dispatching actions is similar to simulations with random delays between 0 and 60 or 0 and 120 minutes.

Dispatching actions could also be observed per day and average per day. As an example, dispatching action diverted and stopped in the station is shown in Fig. 6.

Results show that the number of dispatching actions or the number of trains which are diverted and stopped in stations varies from 20 to 50 but the average number of actions is around 33. Similar variations are observed for the other two dispatching actions, the number of trains which are stopped on the main track in stations varies from 4 to 36 and the average number of stops is 17. The number of trains which are diverted in stations is between 8 and 43 with an average number of trains which are diverted being 23.

Another comparison was performed by analysing average delays per day and the total average number of delays per day. Same random arrival timetable for the 30-day period was used. Results show variations in average delays per day from 7 to 12 minutes but total average delays are around 9 minutes. Similar results are obtained when stations are observed separately, average delays per day vary from 6 to 16 minutes, but total average delays are around 9 minutes in all three stations.

Table 2 Number of dispatching actions in the 30-day period

Dispatching action	Station	Delay intervals / min						
		0 - 10	10 - 20	20 - 30	0 - 30	0 - 60	0 - 120	Random arrivals
Stopped in station	Črnotiče	148	150	148	137	170	201	204
	Hrastovlje	88	82	87	78	154	151	149
	Rižana	77	48	65	61	137	174	166
Diverted in station	Črnotiče	217	207	227	180	178	246	239
	Hrastovlje	105	101	113	101	237	251	228
	Rižana	102	71	85	86	149	180	181
Stopped and diverted in station	Črnotiče	379	297	278	278	337	382	344
	Hrastovlje	294	273	293	286	390	422	388
	Rižana	185	159	173	167	217	243	230

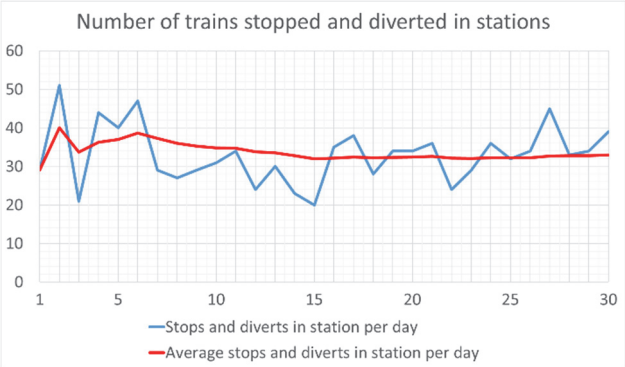

Figure 6 Number of trains stopped and diverted in stations

6 CONCLUSION

High-level Petri nets are a powerful tool for creating models of railway infrastructure. Petri nets can model the processes that include choice, iteration, and concurrent execution. Presented HLPN model for the single-track railway line is flexible and easily adaptable as it is based on a modular principle, where modules are created for any specific section of railway infrastructure. Deadlock avoidance during simulation is addressed and managed in a dispatching logic module. Deadlock avoidance problem is identified by implementing the train dispatching logic directly to a simulation model of a single track railway line. Dispatching logic is implemented by using Petri nets ability to model the concurrent behaviour of distributed systems.

Results obtained from an HLPN simulation are compared to the results from commercial railway software for a selected rail section. Analysis of the results shows that, in case of train meetings, results show a significant number of stops of opposing trains in front of entrance signals in all stations, which in turn prolongs stops of trains already in the station. Reasons lay in the inability to create entrance routes from both directions in stations due to conflicts with route overlap. Route overlap is part of the entrance route behind the exit signal which is reserved in case the train does not stop in front of the exit signal. After train enters the track in the station, route overlap is active 90 seconds in which time that part of the track could not be used for another route. On this line, route overlaps in all three stations lead to open track and permit the creation of entrance routes from opposing direction. Since route overlaps must exist for safety reasons the best solution is to build short dead-end tracks behind exit signals which will be used just for route overlaps. That solution will enable simultaneous entrances in the station from opposing directions which will shorten delays and number of unnecessary stops.

Comparing it to other railway simulation tools, the HLPN model has an advantage as it can provide good modelling power and analysability. This is used to extend the railway simulation model with a decision support tool designed as an integral part of the simulation model. The hierarchical structure of the HLPN model enables a dynamical response to a train routes conflict that appears during a simulation run. The results of the HLPN are therefore affected by a decision on train dispatching generated and enable experimentation with simulation model with various nondeterministic input data (timetables). Simulation of the same section by HLPN with dispatching logic gives better results comparing the number of train stops and waiting times.

The proposed simulation modelling technique should be of interest to a readership focusing on railway engineering and railway simulation. In practice, the proposed HLPN railway simulation model with dispatching actions could be used by planners to model specific and complex railway operation systems. The HLPN model enables experimentation with alternatives and can be used for extensive testing in various initial conditions with stochastic data where a decision on train paths is dynamically determined during simulation by train dispatching logic.

7 REFERENCES

[1] Rozenberg, G. & Jensen, K. (1991). High-level Petri nets: theory and application. Berlin: Springer-Verlag.

[2] Jensen, K. (1990). Coloured Petri nets: a high level language for system design and analysis. Aarhus: Aarhus University, Computer Science Department. https://doi.org/10.1007/978-3-642-84524-6_2

[3] He, X. & Murata, T. (2005). High-Level Petri Nets-Extensions, Analysis, and Applications. The Electrical Engineering Handbook, 459-475. https://doi.org/10.1016/b978-012170960-0/50035-9

[4] Yang, B. & Li, H. (2018). A novel dynamic timed fuzzy Petri nets modeling method with applications to industrial processes. Expert Systems with Applications, 97, 276-289. https://doi.org/10.1016/j.eswa.2017.12.027

[5] Janssens, G. K., Caris, A., & Ramaekers, K., (2009). Time Petri nets as an evaluation tool for handling travel time uncertainty in vehicle routing solutions. Expert systems with applications, 36, 5987-5991. https://doi.org/10.1016/j.eswa.2008.07.001

[6] Xing, K., Wang, F., Zhou, M. C., Lei, H., & Luo, J. (2018). Deadlock characterization and control of flexible assembly systems with Petri nets. Automatica, 87, 358-364. https://doi.org/10.1016/j.automatica.2017.09.001

[7] Yue, H., Xing, K., Hu, H., Wu, W., & Su, H. (2016). Petri-net based robust supervisory control of automated

manufacturing systems. Control Engineering Practice, 54, 176-189. https://doi.org/10.1016/j.conengprac.2016.05.009
[8] Liu, Z., Liu, Y., Cai, B., Li, J., & Tian, X. (2017). Reliability analysis of multiplex control system of subsea blowout preventer based on stochastic Petri net. Tehnički vjesnik, 24(1), 7-14. https://doi.org/10.17559/TV-20130502140334
[9] Cheng, Y. H. &Yang, L. A. (2009). A Fuzzy Petri Nets approach for railway traffic control in case of abnormality: Evidence from Taiwan railway system. Expert systems with applications, 36, 8040-8048. https://doi.org/10.1016/j.eswa.2008.10.070
[10] Meng, X., Jia, L., & Xiang, W. (2018). A Petri Net Model of Train Operation Simulation for Harmonizing Train Timetables of Neighbor Dispatching Sections. Promet - Traffic & Transportation, 30(6), 647-660. https://doi.org/10.7307/ptt.v30i6.2713
[11] Malavasi, G. & Ricci, S. (2002). Railway traffic simulation by means of a Petri Net model. In Computers in Railways VIII, 407-415. Southampton: WIT Press.
[12] Potekhin, A. I., Branishtov, S. A., & Kuznetsov, S. K. (2016). Discrete-Event Models of a Railway Network. Automation and Remote Control, 77, 344-355. https://doi.org/10.1134/s0005117916020107
[13] Fay, A. (2000). A fuzzy knowledge-based system for railway traffic control. Engineering Applications of Artificial Intelligence, 13(6), 719-729. https://doi.org/10.1016/s0952-1976(00)00027-0
[14] Zhuang, H., Feng, L., Wen, C., Peng, Q., & Tang, Q. (2016). High-Speed Railway Train Timetable Conflict Prediction Based on Fuzzy Temporal Knowledge Reasoning. Engieering, 2, 366-373. https://doi.org/10.1016/j.eng.2016.03.019
[15] Fanti, M., Giua, A., & Seatzu, C. (2006). Monitor design for colored Petri nets: An application to deadlock prevention in railway networks. Control Engineering Practice, 14(10), 1231-1247. https://doi.org/10.1016/j.conengprac.2006.02.007
[16] Pachl, J. (2011). Deadlock Avoidance in Railroad Operations Simulations, Transportation Research Board 90th Annual Meeting, Washington DC,23-27 Jan 2011. Washington DC: Transportation Research Board
[17] Milinković, S., Marković, M., Vesković, S., Ivić, M., & Pavlović, N. (2013). A fuzzy Petri net model to estimate train delays. Simulation Modelling Practice and Theory, 33, 144–157. https://doi.org/10.1016/j.simpat.2012.12.005
[18] Murata, T. (1989). Petri nets: Properties, analysis and applications, Proceedings of the IEEE, 77(4), 541-580. https://doi.org/10.1109/5.24143

Contact information:

Dušan JEREMIĆ
(Corresponding author)
University of Belgrade, Faculty of Transport and Traffic Engineering,
11000 Belgrade, Vojvode Stepe 305, Serbia
E-mail: jeremicd@gmail.com

Sanjin MILINKOVIĆ, PhD
University of Belgrade, Faculty of Transport and Traffic Engineering,
11000 Belgrade, Vojvode Stepe 305, Serbia
E-mail: s.milinkovic@sf.bg.ac.rs

Sandra KASALICA, PhD
High Railway School of Vocational Studies,
11000 Belgrade, Zdravka Čelara 14, Serbia
E-mail: sandra.kasalica@gmail.com