A Railway Scheduling Method using Probabilistic Model Checking

Mohammadsadegh Mohagheghi, Anahita Khademi

Abstract: Trains scheduling is an important problem in railway transportation. Many companies use fixed train timetabling to handle this problem. Train delays can affect the pre-defined timetables and postpone destination arrival times. Besides, delay propagation may affect other trains and degrade the performance of a railway network. An optimal timetable minimizes the total propagated delays in a network. In this paper, we propose a new approach to compute the expected propagated delays in a railway network. As the main contribution of the work, we use Discrete-time Markov chains to model a railway network with a fixed timetable and use probabilistic model checking to approximate the expected delays and the probability of reaching destinations with a desired delay. We use PRISM model checker to apply our approach for analyzing the impact of different train scheduling in double line tracks.

Keywords: Discrete-time Markov chains, Probabilistic model checking, Railway transportation, Train scheduling.

I. INTRODUCTION

Train scheduling is one of the most important problems in the railway transportation, which influences other industry parts of a country [1]. It is not easy to predict delays in a complex railway system. Engine failure, human behavior, and external environment cause unpredictable delays. In addition, a train may propagate its delays to other trains in a heavy load condition [1], [2]. It is critical for a railway system to have optimal scheduling which minimizes the propagated delays. A bad scheduling increases the total delays which also increases the cost of using trains for transportation. A standard approach for modeling train scheduling is to consider it as a stochastic process [3], [4]. Discrete-time Markov chains (DTMCs) can be used to model and analysis the stochastic behavior of several trains in a railway scheduling system. In this case, statistical and technical information are used to approximate the failure rate and the probability of delays of each individual train. In this paper, we propose a method to model the trains scheduling with DTMCs. We rely on fixed trains timetabling and the proposed method is used to approximate the expected delays and the probability of reaching destinations when the networks tolerate a limited delay in 24 hours. We consider double-track lines. Our method converts a train scheduling to a probabilistic model checking problem and uses the PRISM tool [5] to study the related properties. To the best of our knowledge, this work is the first one that uses probabilistic model checking to approximate the propagated delays for a timetable scheduling in a railway system.

There are several standard algorithms for off-line timetabling problems. These algorithms are reviewed and compared in [1], [6], [7]. To improve the performance of the standard algorithms several new methods are proposed in [1]. A genetic algorithm is proposed in [6] to find a (near) optimal solution for the train timetabling problem. The railway environment is considered as a stochastic problem in [3]. In this case, the problem is converted to an integer programming one and numerical methods are used to compute the best solution. Probabilistic model checking is used in several previous works for modeling the trains scheduling problem. In [4] Markov decision processes are used to model the trains in a metro network and probabilistic model checking is used to approximate the expected time to reach a stable condition in a delayed line. In [8] probabilistic model checking is used for train rooting to determine the expected cost and risk of several paths in a railway system.

II. A REVIEW OF PROBABILISTIC MODEL CHECKING

Model checking is a formal approach for verifying the correctness and reliability of a given system [9]. A model checker is a software tool used to systematically compute the desired properties. Probabilistic model checking is one variant for verification of systems with stochastic and probabilistic behaviors. In this approach, a probabilistic structure (such as DTMC of Markov decision process) is used for modeling the system and a probabilistic logic is used for specification. In this section, we review DTMCs and briefly explain how a qualitative property is proposed and computed in an automatic approach. More details are available in [9], [10].

Definition 1. Discrete-time Markov chain (DTMC)
A DTMC is defined as a tuple of the form $D=\{S,s_0,G,P,R\}$ where S is a finite set of states, $s_0 \in S$ is the initial state, $G \subset S$ is the set of goal states, $P: S \times S \to [0,1]$ is the transition probability function and $R: S \times S \to R^{\geq 0}$ is a reward function.

DTMCs are widely used to model the behavior of systems with some stochastic and deterministic aspects [9]. The main advantage of DTMCs is their memory-less structure, i.e. each state shows the current status of the system independent of the history of its previous states. The system starts from the initial state s_0. The behavior of the system is traced by a sequence of

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
Mohammadsadegh Mohagheghi*, Department of Computer science, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran. Email: mohagheghi@vru.ac.ir
Anahita Khademi, Department of Computer science, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran. Email: anahitakhademi92@gmail.com

states (called path) of the form $s_0 s_1 s_2 \dots$ where each state s_{i+1} is randomly selected from s_i with probability $P(s_i, s_{i+1})$. A transition of D is any pair (s_i, s_j) of states where $P(s_i, s_j) > 0$. For a transition (s_i, s_j) the reward $R(s_i, s_j)$ is mapped that shows the value that is gained by the system with this transition. Two main classes of properties that are proposed against DTMCs are reachability probabilities and expected accumulated rewards. In reachability probabilities, the probability of finally reaching one of goal states is computed. In expected accumulated rewards, the expectation of accumulated rewards among those paths that are reached to some of the goal states is considered. A model checker uses numerical commutations to determine reachability probabilities or expected accumulated rewards. A standard approach to define a DTMC model is to use a high-level descriptive language (PRISM language for example). A model checker maps the described model to its corresponding DTMC. In Section 4 we explain how to model a railroad scheduling problem with the PRISM language.

III. Problem Definition

In this paper, we consider the fixed timetable train scheduling problem [1]. An important property of a good timetable scheduling is to minimize delay propagations. We consider a double line track system where trains from each side use one of two lines exclusively. Fig. 1. shows the schema of a double line track. There are two source and destination stations and several stations with known distances exist between them. In a timetable scheduling, the departure time of each train is determined. In a double track system, the timetable of each side is independent of the other side. For simplicity, we consider one side and we suppose that each train travels on a fixed speed. To avoid collision, a minimum distance between each two consecutive trains should be considered. If one train stops in a line (between two stations), the other trains need to stop to avoid any collision. This case results in delay propagation between trains [1]. An optimal timetable scheduling should minimize the expected delay propagation. Ideally, departure times should be as far as possible to minimize this value. However, good train scheduling should also satisfy high demands during peak hours. In this case, we are interested in approximating the propagated delays for a given timetable scheduling.

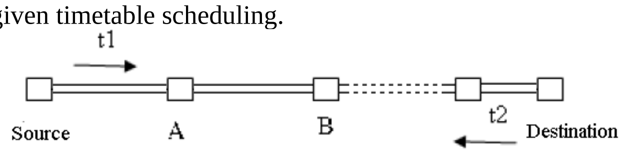

Fig. 1. A schema of a double-track railway

IV. The Proposed Method

We use DTMCs to model the trains scheduling problem. A model includes the location of each station and the movement of trains between stations. Although railways are continuous systems in space and time, we need some simplification for modeling them in DTMCs [4]. We consider every five minutes as a time unit in our model. Also, we suppose that each train travels at a fixed speed and its dwell time in each station is at least one time unit. As a result, the model is defined as a sequence of locations where each train moves from a location to its successor location in one time unit.

A. The general approach for modeling a railway scheduling problem

To model delays in a trip, we consider a parameter α as the probability of failure after each time unit. A train stays in its location in case of failures. For simplicity, we consider one time unit delay for recovering the failure. After recovery, the train moves to the next location. Train failures have Markov property that means the probability of failures is not changed after each recovery and also does not depend on the travel time. The locations between each two consecutive stations define a line. As an example, there are two lines between the stations A and B in Fig. 1 if we consider it as a double. To model a timetable, we use the departure time of each train in the model in 24 hours. For example, if the departure time of a train is 2 hours after the starting point of the system, we set its departure time to 24, which is 24 time units (24 * 5 minuts) after start point.

B. Defining the DTMC model for train scheduling in PRISM

To analyze the underlying properties of a timetable for a railway we define the components of the system in the PRISM language. Any DTMC model in PRISM is defined as several modules where each module has one or some integer variables in the defined range of values. The overall DTMC model is constructed as a synchronous product of the modules. Each state represents a possible valuation for the variables of modules.

Figure 2 presents two modules and the initial definition of a DTMC model for our problem. In this problem, we define one module for each train (train1, train2, …) and also a module for the timer. For each time unit, the value of the timer is incremented by one that shows the elapse of 5 minutes. For each module *traini* a *loci* variable is considered to determine its location, a *delayi* variable to show the intrinsic delays of the train from its departure and a *dli* variable to show the propagated delay reached from other trains. Note that *delayi* is incremented if the train has a failure. To simplify, we limit the maximum value for each *delay* variable. Otherwise, a delay variable may have any large value which is not so realistic. The PRISM source code of a case study model with the related log files is available in: *https://github.com/sadeghrk2/prismrailways.*

C. Modeling order of trains

Note that no train can pass the others. As a result, the order of trains is important in our model. If several trains dwell in a station they should leave it in their initial order, i.e. priority of the successor trains is more than the priority of its predecessor. To avoid collision, we consider several conditions for transitions of each module: if the distance between each two consecutive trains is less than two time unit, the second train (which is behind the first one) had to stop until the first one leaves its location. In this case, the *dli* value of the second trains is incremented. Finally, if a train arrives at destination, its location does not change forever.

In our model, we consider four stations between source and destination and we suppose that there are 12 locations between every two consecutive stations. We define 6 train modules and the default value for α is 0.05 that shows the probability of failure after each time unit. To model the timetable, several constant values are defined that show the departure time of each train. To analyze the impact of several timetables we define a distance parameter that determines distances between each two departure times.

```
dtmc
//definition of a time table for 6 trains :
const distance;
const st1 = 2;
const st2 = distance + 2;
const st3 = 2 * distance + 2;
const st4 = 3 * distance + 2;
const st5 = 4 * distance + 2;
const st6 = 5 * distance + 2;
//definition of modules
module timer
 t : [0..288];
 [tick] t < 250 -> (t' = t + 1);
 [tick] t = 250 -> (t' = t);
endmodule

module train1
 loc1 : [0..60] init 0;
 delay1 : [0..20];
 [tick] t < st1 & loc1 = 0 -> (delay1' = 0);
 [tick] t = st1 & loc1 = 0 -> (loc1' = 1);
 [tick] loc1 >= 1 & loc1 < 12 & delay1 < 20 -> .95:(loc1' = loc1 + 1) +
.05:(delay1' = delay1 + 1);
 [tick] loc1 >= 1 & loc1 < 12 & delay1 = 20 -> (loc1' = loc1 + 1);
 [tick] loc1 = 12 -> (loc1' = 13);
 [tick] loc1 >= 13 & loc1 < 24 & delay1 < 20 -> .95:(loc1' = loc1 + 1)
+ .05:(delay1' = delay1 + 1);
 [tick] loc1 >= 13 & loc1 < 24 & delay1 = 20 -> (loc1' = loc1 + 1);
 [tick] loc1 = 24 -> (loc1' = 25);
 [tick] loc1 >= 25 & loc1 < 36 & delay1 < 20 -> .95:(loc1' = loc1 + 1)
+ .05:(delay1' = delay1 + 1);
 [tick] loc1 >= 25 & loc1 < 36 & delay1 = 20 -> (loc1' = loc1 + 1);
 [tick] loc1 = 36 -> (loc1' = 37);
 [tick] loc1 >= 37 & loc1 < 48 & delay1 < 20 -> .95:(loc1' = loc1 + 1)
+ .05:(delay1' = delay1 + 1);
 [tick] loc1 >= 37 & loc1 < 48 & delay1 = 20 -> (loc1' = loc1 + 1);
 [tick] loc1 = 48 -> (loc1' = 49);
 [tick] loc1 >= 49 & loc1 < 60 & delay1 < 20 -> .95:(loc1' = loc1 + 1)
+ .05:(delay1' = delay1 + 1);
 [tick] loc1 >= 49 & loc1 < 60 & delay1 = 20 -> (loc1' = loc1 + 1);
 [tick] loc1 = 60 -> (loc1' = 60);
endmodule
```

V. Experimental Results

To compute the properties of the DTMC models of timetable scheduling problems, we use PRISM 4.5 for implementing and running each model. Because of the complexity of the defined models, the standard iterative methods cannot be used for probabilistic model checking. Instead, we use the statistical approach that is based on the Monte-Carlo simulation [11]. We set the parameters of statistical model checking to have precise values in the first two meaningful digits. Several properties can be defined to compare the impact of different timetables on the performance of a railway system. We first consider the expected value of total intrinsic delays which is mathematically defined as

$$E(\sum_{i=1}^{6} delayi) \qquad (1)$$

Note that in this case, we do not consider the propagated delays and only compute the expected value of total delays. We consider several values for the parameter α and run the experiments for these values. The results of these experiments are proposed in Fig. 3.

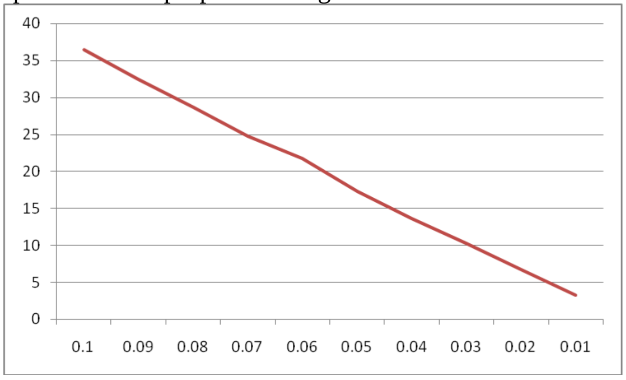

Fig. 3. The value of Equation (1) for different values of α

The horizontal axis shows the value for the parameter α and the vertical axis shows the value of equation (1). This figure shows that there is approximately a linear relationship between the value of α and Equation (1). For example, when α is 0.1 we expect to have about 36 time unit delay for 6 trains. Note that the value of this parameter depends on the technical properties of trains and the environment of the railway.

The second property that is studied is to approximate the expected propagated delays for a given timetable. This value is defined as:

$$E(\sum_{i=1}^{6} dli) \qquad (2)$$

Where *dli* is used for the propagated delay of train *i*. For simplicity, we suppose that the distances between each two consecutive trains are the same and we set $\alpha = 0.05$. Figure 4 shows the values of equation (2) for different values of the *distance* parameter.

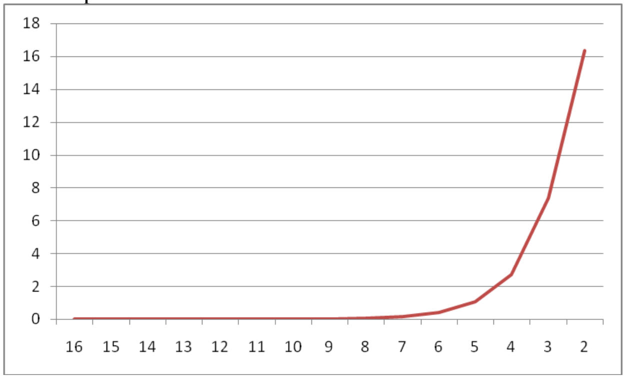

Fig. 4. The values of Equation (2) according to the values of distance

The vertical axis shows the values for Equation (2) and the horizontal axis shows the values for *distance*. The results show that the propagated delays are negligible when the distance between departure times are more than 7 time units. On the other side, the propagated delays increase when the distance between departure times are small.

These results can be used to have a tradeoff between high demand in peak hours and the need to have small propagated delays.

As the third property, we are interested to compute the probability of reaching a destination station while the total propagated delays are less than two time units. This probability is computed for the case that all trains have finished their travel. The results of the experiments for this property are proposed in Fig. 5.

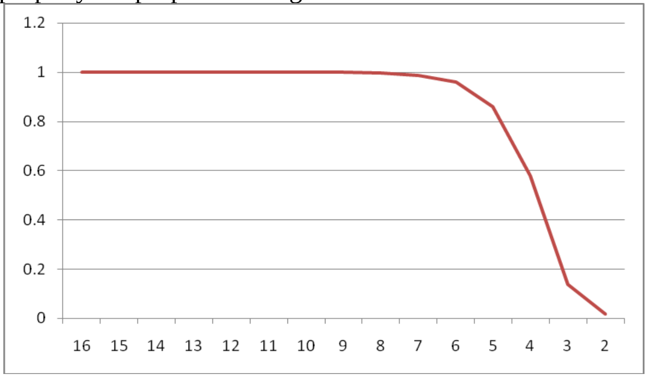

Fig. 5. Probability of reaching destination while total propagated delays are less than or equal to two time units.

The results of Figure 5 (similar to the results of Figure 4) show that the probability of reaching destination while the propagated delays are less than 3 time units is approximately one. This means that almost surely, all trains can reach to the destination while the total propagated delays is less than 3. To have a better analysis for the case of heavy load, where the value of *distance* is small, we also consider the probability of reaching destination station while the total delay is less than or equal to 6 or 10 time units. Fig. 6 shows the results for these experiments.

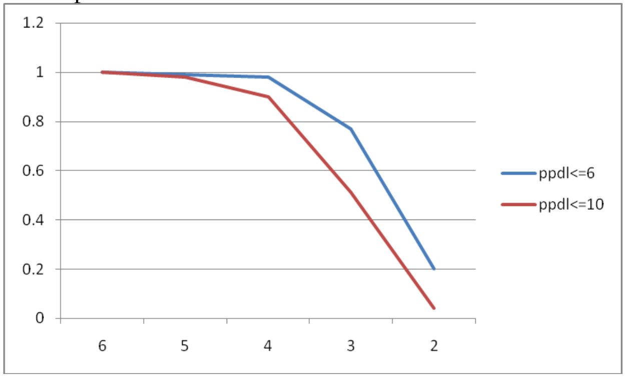

Fig. 6. Probabilities of reaching destination for two bound on the total propagated delays.

In this Figure we use “ppdl” for the total propagated delay. Again, the horizontal axis shows the value of *distance* in a given time table. Although Figure 4 and 5 propose that the performance of the system is decreased when *the distance* is 5 or 4, the results of Figure 6 show that these two cases are much better than the cases where *distance* is 3 or 2.

VI. Conclusion and Related Works

In this paper, we propose a new approach for modeling and analyzing the trains scheduling problem. We consider fixed time-table scheduling and use probabilistic model checking to approximate the expected propagated delays or probabilities of reaching destination with a limited number of propagated delays. We use PRISM model checker to analyses the related models. The results show that with the assumptions of this paper, the performance of a railway system degrades when the distance between any two departure times is less than 7 time units. For the future, one can consider a case study and apply the proposed method to study its performance. Expanding the system to a network of railroads and considering single-track lines are two other directions for future researches.

REFERENCES

1. D'Ariano, Andrea. "Improving real-time train dispatching: models, algorithms and applications.".D. thesis, Delft University, 2008 .
2. Acuna-Agost, Rodrigo, Philippe Michelon, Dominique Feillet, and Serigne Gueye. "SAPI: Statistical Analysis of Propagation of Incidents. A new approach for rescheduling trains after disruptions." *European Journal of Operational Research* 215, no. 1 (2011): 227-243.
3. Meng, Lingyun, Xiaojie Luan, and Xuesong Zhou. "A train dispatching model under a stochastic environment: stable train routing constraints and reformulation." *Networks and Spatial Economics* 16, no. 3 (2016): 791-820.
4. Bertrand, Nathalie, Benjamin Bordais, Loïc Hélouët, Thomas Mari, Julie Parreaux, and Ocan Sankur. "Performance Evaluation of Metro Regulations Using Probabilistic Model-checking." In *International Conference on Reliability, Safety, and Security of Railway Systems*, pp. 59-76. Springer, Cham, 2019.
5. Marta Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In Proc. 23rd International Conference on Computer Aided Verification (CAV’11), volume 6806 of LNCS, pages 585-591, Springer, 2011.
6. Törnquist, Johanna. "Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms." In 5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.
7. Tormos, P., A. Lova, Federico Barber, L. Ingolotti, Montserrat Abril, and Miguel A. Salido. "A genetic algorithm for railway scheduling problems." In Metaheuristics for scheduling in industrial and manufacturing applications, pp. 255-276. Springer, Berlin, Heidelberg, 2008.
8. Soeanu, Andrei, Mourad Debbabi, Dima Alhadidi, Makram Makkawi, Mohamad Allouche, Micheline Bélanger, and Nicholas Léchevin. "Transportation risk analysis using probabilistic model checking." Expert Systems with Applications 42, no. 9 (2015): 4410-4421.
9. Baier, Christel, and Joost-Pieter Katoen. Principles of model checking. MIT press, 2008.
10. Hahn, Ernst Moritz, Arnd Hartmanns, Christian Hensel, Michaela Klauck, Joachim Klein, Jan Křetínský, David Parker, Tim Quatmann, Enno Ruijters, and Marcel Steinmetz. "The 2019 comparison of tools for the analysis of quantitative formal models." In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 69-92. Springer, Cham, 2019.
11. Forejt, Vojtěch, Marta Kwiatkowska, Gethin Norman, and David Parker. "Automated verification techniques for probabilistic systems." In International School on Formal Methods for the Design of Computer, Communication and Software Systems, pp. 53-113. Springer, Berlin, Heidelberg, 2011.

AUTHORS PROFILE

Mohammadsadegh. Mohagheghi Received his Ph.D in computer science from University of Tabriz in 2019, MS in computer science from Sharif University of technology in 2008 and BSc in software engineering from Shahidbeheshty University in 2006. He is currently a faculty member of computer science in Vali-e-asr University of Rafsanjan, Iran. His main research interests include formal verification of stochastic and real-time systems, probabilistic model checking and machine learning. He has an experience of study in several courses of computer science including: Theory of computations, Introduction to programming, Algorithm design, Compiler design, Programming languages, Advanced programming and Data base and Artificial Intelligence.

Anahita Khademi Received her MS in Artificial intelligence from Yazd university in 2016 and her BS from Vali-e-asr University of Rafsanjan in 2011. Her main research interests include artificial intelligence, robotics and machine learning. She is currently a part-time lecturer in Department of Computer science, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran.