IMS Project Report

Tomáš Brablec (xbrabl04) **Pavel Štarha (xstarh04)**

Contents

1 Introduction

This work attempts to create a simulation that is able to find the optimal policy for scheduling trains. Specifically, the goal is to find a policy for delaying trains to allow passengers to transfer from a delayed train to another. This is a problem without a simple solution since the optimal policy will depend on the specific conditions of the analyzed connections. The resulting simulation model can be configured for a range of situations.

1.1 Authors

This work was created by the project authors, based on studying route plans in [1] and [2]. Other sources include information about connecting train policies in [3] and [4], and timetable creation policies from [5]. Statistics of train delays are extracted from [6].

1.2 Validity of the Model

Model configurations used in simulations use real data gathered from sources described above. Additionally, validity of the model is checked firstly by simulating simple scenarios and checking that the results align with both our expectations and real-world data (see Section 5.2). Secondly, a real-world track is simulated (see Section 5.2.5) and checked that the results of the simulation align with data from [6].

2 Problem Analysis

In our research we decided to focus on policies of waiting for delayed trains used by the Czech Railways. The current policy defines multiple tiers of trains [5], where only a train of lower tier will wait on a train of a higher tier, not the other way around.

The current policy states that the train will wait only for such a time, so that it will leave the station with its own delay no greater than 5 minutes, as stated in [4]. This means that a train which already accumulated a delay of 5 minutes will not wait for another train in any situation. Note that this is not equivalent to the policy of simply waiting the maximum of 5 minutes in each station, because this would allow for accumulation of delay greater than 5 minutes over multiple stations.

Our goal is to try out the behavior of similar policies on a simple model and to check whether we can find one that performs as well or better than the current one.

2.1 Our Approach

Our conceptual model is represented using a Petri Net with some informal notation added on top. This is done to simplify the model when expressing more complicated relationships. Other models of railway systems, such as the one in [7], employ extensions of Petri Nets, such as their colored or fuzzy variants.

Our simulation model is implemented in C++ using the SIMLIB [8] simulation library. The choice of this library is based on familiarity with its API (explained in [9]) and its speed. Our simulation experiments often required millions of model time units to produce clean results, which would be hard to achieve in a language with more runtime overhead than C++.

2.2 Origin of our Method

Initially, we planned to create a full model of a railway system comparable to [7]. However, the paper focuses on scheduling railway traffic in general, with respect to safety and liveness properties. This is not the focus of our work, therefore we decided to omit most of the details while still satisfying the basic rules of railway traffic outlined in [7]. Mainly that each track

section can be occupied by at most one train at a time. Exactly how this is satisfied in our model is described in Section 3.

3 Conceptual Model

Our conceptual model describes a railway track with a series of stations. This track (from now referred to as the *main track*) is served by a single train (referred to as the *connecting train*). There are two types of stations. In the first type, only *direct passengers* board the train. The connecting train rides from one end station to another and back in a loop.

Direct passenger comes to a station at the moment of scheduled arrival of the connecting train, boards the train, and then leaves the train at another station (their destination). Their travel time is counted from the scheduled (not actual) arrival of the connecting train to the moment of arriving into their destination.

The second type of station is a *transfer station*, where, in addition to direct passengers, *independent trains* finish their journey. The movement of these trains themselves is not simulated since it does not affect the resulting data, the only information about independent trains is their delay at the moment of arrival to the transfer station. The scheduled (not actual) arrival of an independent train is synchronized to the arrival of the connecting train. Therefore, if the independent train arrives without a delay, all the passengers of the independent train will be able to transfer to the connecting train.

The independent train carries *transfer passengers*, which arrive to the transfer station in a (possibly delayed) independent train, then wait for the soonest connecting train heading in the direction of their destination station, board the train and continue the same way as direct passengers. Passengers arriving in the independent train to the transfer station and then not continuing in the connecting train are not simulated, since their total time of travel is not affected by the policy of waiting on delayed trains.

The destinations of both direct and transfer passengers are generated randomly along the main track. The only rule that applies is that the destination station is different from the origin station (transfer station for transfer passengers).

As described in Section 2, only the connecting train can wait for delayed independent train (hence the naming), not the other way around. Therefore, the delay of independent train can be generated independently of any other parameter based on a distribution extracted from real data.

In our model, a track section mentioned in [7] is simply the section of a track between two neighboring stations. Therefore, the safety property is satisfied automatically by simulating only one train on the main track, and ensuring that the delay of the independent trains does not exceed its period of repetition. That would mean that one train caught up with another on a single track.

3.1 Model Description

Figure 1 displays a schematic diagram of an example configuration of our model. Number and properties of stations and independent trains is fully configurable in our simulation model, so that we can simulate different situations using a single model. The parameter P at each station describes the number of direct passengers departing from the station each time the connecting train is scheduled to arrive. T_{next} is the time of ride into the next station in minutes. Note that T_{next} of the last station is zero, since there is no next station.

Independent trains are marked with a dot, their parameter P determines the number of transfer passengers travelling in the train. Parameter D represents the delay of the train and T_{mul} represents the number of connection train periods it takes for this train to arrive. The *connection train period* T_c is time it takes the connection train to do one full cycle over the track, from the starting station to the opposite station and back. The value $T_{\text{mul}} = 4$ means that the independent train will arrive every 6 hours (connection train period is 90 minutes in Figure 1).

Figure 2 shows a Petri Net model of a single transfer station with an independent train. Notice that the *maximum wait time* T_w is reduced by the current *connecting train delay* T_d. This means that if the current delay is equal to the wait time, the connecting train will not wait for any other train. This models the property described in Section 2.

Figure 3 is a Petri Net model of a simple station (without an independent train). It shows that the train will not spend any time except the *boarding time* in the station since there is no other train to wait for.

3.2 Model Diagrams

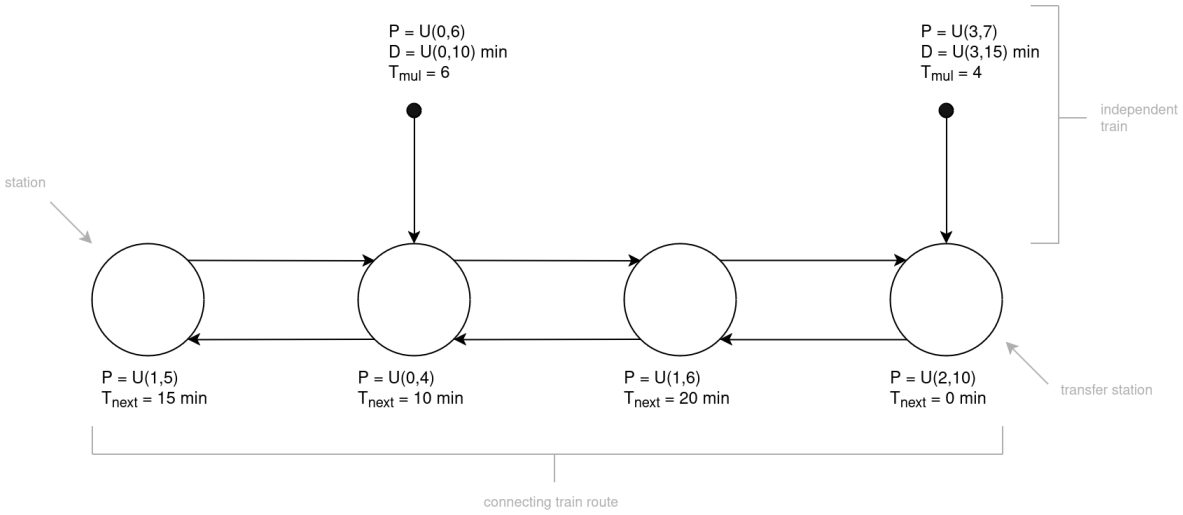

Figure 1: Schematic model of an example track with four stations

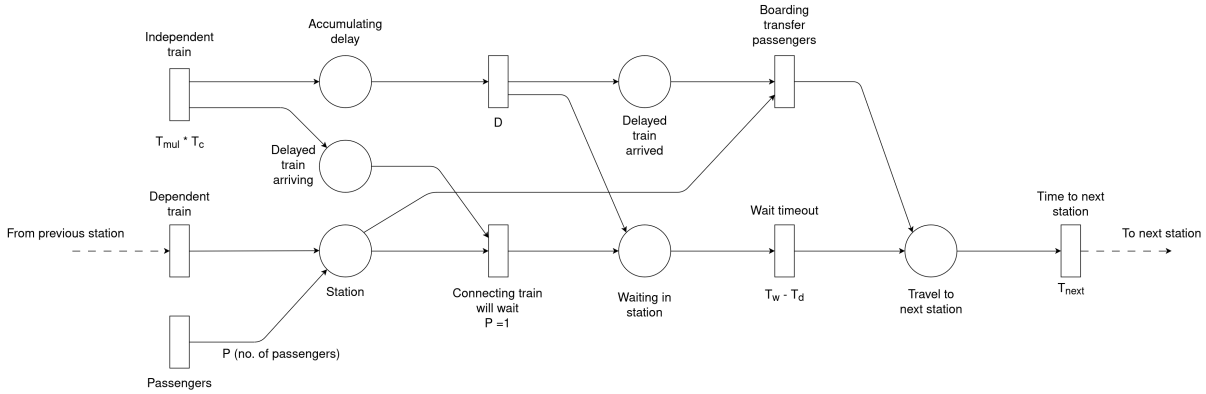

Figure 2: Petri Net model of a transfer station with an independent train and direct passengers

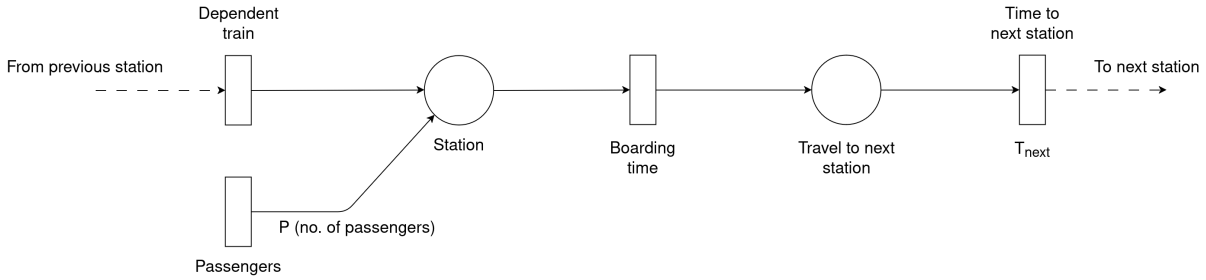

Figure 3: Petri net model of a simple station with direct passengers

4 Simulation Model

The simulation model is implemented generically with respect to the number and properties of stations. The model time (variable `Time` in SIMLIB) is counted in minutes.

4.0.1 Station

Each station is represented by the following structure (`station.h`):

```
struct Station {
  Facility is_arriving;
  Queue waiting_passengers_left;
  Queue waiting_passengers_right;

  bool has_independent_train;

  IntGenerator transfer_passengers;
  IntGenerator direct_passengers;

  int repeat_period_multiplier; // T_mul
  int time_to_next_station;
  DoubleGenerator get_delay;
}
```

The types `IntGenerator` and `DoubleGenerator` represent functions that return `int` and `double` respectively, and they are used to represent distributions, such as `Uniform` with given parameters, or custom distribution.

Facility `is_arriving` is seized by the process `IndependentTrain` (`independent_train.h`) at the moment of scheduled arrival to the transfer station and is released at the moment of actual (delayed) arrival. The facility is checked by the connecting train at the moment of arrival into a station, and if it is seized, the train begins the wait.

The queues `waiting_passengers_left` and `waiting_passengers_right` are used by both types of passengers to wait till the arrival of the connecting train. The direction is determined automatically according to the destination station (passengers will not board the connecting train and ride it to the end station and back just to get to a station one place over).

The parameter `has_independent_train` determines the type of station. For code simplicity, both types of stations are represented using the same structure.

Functions `transfer_passengers` and `direct_passengers` return the number of transfer and direct passengers, respectively. `get_delay` returns the delay of the independent train when called.

4.0.2 Independent Train

The process `IndependentTrain` is implemented in `independent_train.cpp` and behaves in the following way. When first activated, it activates all its passengers so that they start counting their travel time. Then the train waits for the duration of its delay and activates its passengers again. This represents the arrival into the transfer station.

4.0.3 Connecting Train

The connecting train process (`connecting_train.cpp`) runs in a loop – a single instance is spawned at the start of the simulation and runs until the end. It cycles across the station list from start to end and back, in each station it first activates all leaving passengers, which informs them that they arrived to their destination, then boards all passengers waiting in the stations (from the queue corresponding to the current direction of travel). Finally, it checks whether there is an independent train arriving with delay, and waits for it according to the rules specified in Section 3. Then it continues to the next station.

When the train travels between stations with a delay, it can shorten the delay by the time it would normally spend waiting in the station. It is a common practice for a train to stop in larger stations for more time than is needed for the passengers to board the train. This time can then be used to shorten the delay. This time is constant for all stations (constant `MAX_BOARDING_TIME` in `params.h`). This constant can be derived from real data as the average of boarding times over all stations (for example, for the train Os 4758 it is approximately 1.2 minutes per station, see [1]).

4.0.4 Passengers

There is a single `Passenger` process for both direct and transfer passengers, which performs only three simple actions. When first activated, it logs the time of activation, then passivates itself. This represents waiting for arrival into their transfer station. Direct passengers are reactivated immediately. Then the process enqueues itself into the appropriate direction queue in the transfer station and waits to be dequeued by the connecting train. After being dequeued, it passivates itself again and waits for the activation by the `ConnectingTrain` process at the moment of arrival into its destination.

4.0.5 Parameters

Parameters of the simulation are stored in the file `params.h`. However, due to code structure, the table `stations` which contains the configuration of each station is located in `station.cpp`.

4.1 Correspondence to the Conceptual Model

There is a direct correspondence between the processes `IndependentTrain` and `ConnectingTrain` in the simulation model and in the conceptual model. The processes of the simulation model behave as it is described in Section 3. The parameters in `params.h` also directly correspond to parameters described in Section 3.

5 Simulation Experiments

5.1 Our Approach to Experiments

The purpose of the experiments is to evaluate which maximum wait time (T_w) is optimal for most passengers. To evaluate this, we run simulations with different values of T_w and collect travel times (time from first activation of a passenger to the reaching of final destination). This is done using the `Histogram` class in SIMLIB (`travel_time` in `utils.cpp`), which gives us

the average value of the travel times. The lower this value is, the less time passengers spend travelling, therefore we look for the lowest possible average travel time.

The common approach to the experiments was that we chose a track of a single train, extracted information about the route (travel time between stations, average delays) from [2] and [6], information about independent trains and their delays, and approximated the number of passengers based on the size of the stations. Then we fed this data to the model and evaluated different values of maximum wait time, and plotted the results. In addition to real tracks, we also experimented with fictional tracks with hand-picked parameters, on which we checked that the model behaves as we would expect in certain situations.

5.2 Experiments

5.2.1 No independent trains

In this experiment, we created a track with three stations and no independent trains. All passengers carried by the connecting train are therefore direct passengers. Then we modified the simulation model to give the connecting train a constant delay and disallowed it to reduce the delay. The expected result of this is that with increasing delay, the average travel time will increase in a linear way. The number of direct passengers was set to 25 in each station, and the travel times between stations were set to 20 and 15 minutes.

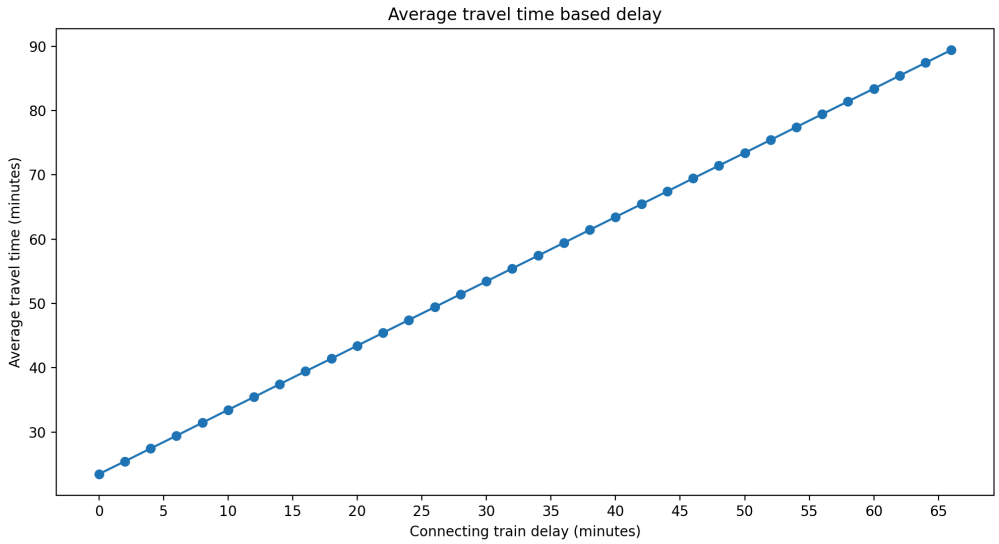

Figure 4: Experiment result: linear growth of average travel time with growing constant delay

5.2.2 Short track

This experiment simulates a scenario where a train waits at each stop for a connecting train, which arrives with a delay generated by the `get_delay` function based on real data from [6] (the delay is less than 15 minutes with a probability of 78%), and the route was set for a short travel time. The best result was achieved when the train did not wait for any independent trains at all because the connecting train period was short (70 minutes), making it unnecessary to wait since the transfer passengers would not wait long for the connecting train to make another loop.

The configuration involved three stations where the train waited for an independent train at each one. All independent trains carried 50 passengers, with 15 direct passengers boarding at each station. The travel time between stations was 20 and 15 minutes respectively, and

the independent trains arrived at the transfer stations with a periodicity of 5th, 4th, and 3rd connecting train period respectively. The boarding time was capped at a maximum of 3 minutes.

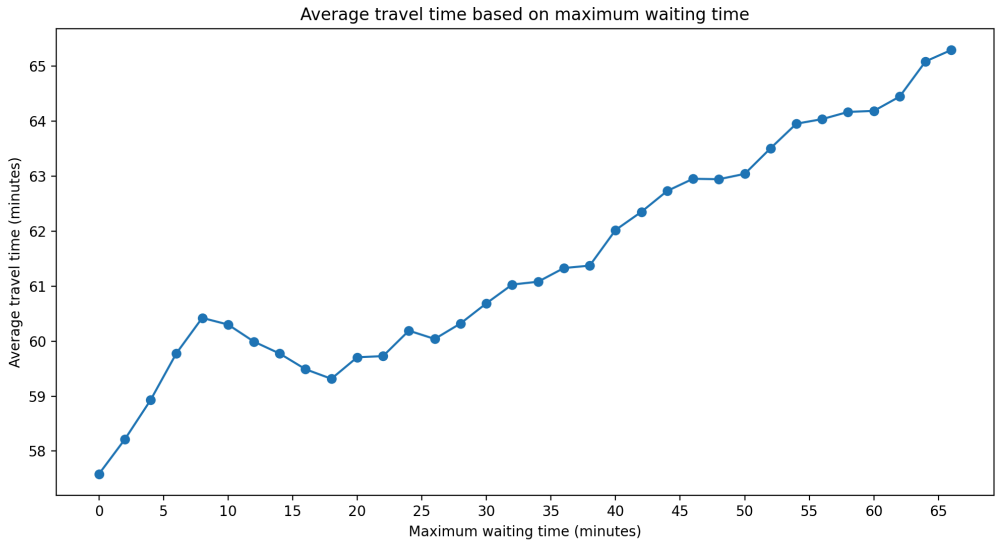

Figure 5: Experiment result showing the average travel time on a short track

5.2.3 Long track

This experiment is based on the previous one, where we added an additional stop to make the main track longer to hopefully get different results from the previous experiment. Between the second and third stations, another stop is added, where an independent train carrying 50 passengers arrives every 4 connecting train periods. 15 direct passengers board the connecting train, and the travel time to the next station was 15 minutes. This experiment demonstrates that as the number of stops (and consequently the connecting train period) increases, it becomes more advantageous to wait for transfer passengers, even at the cost of delaying direct passengers.

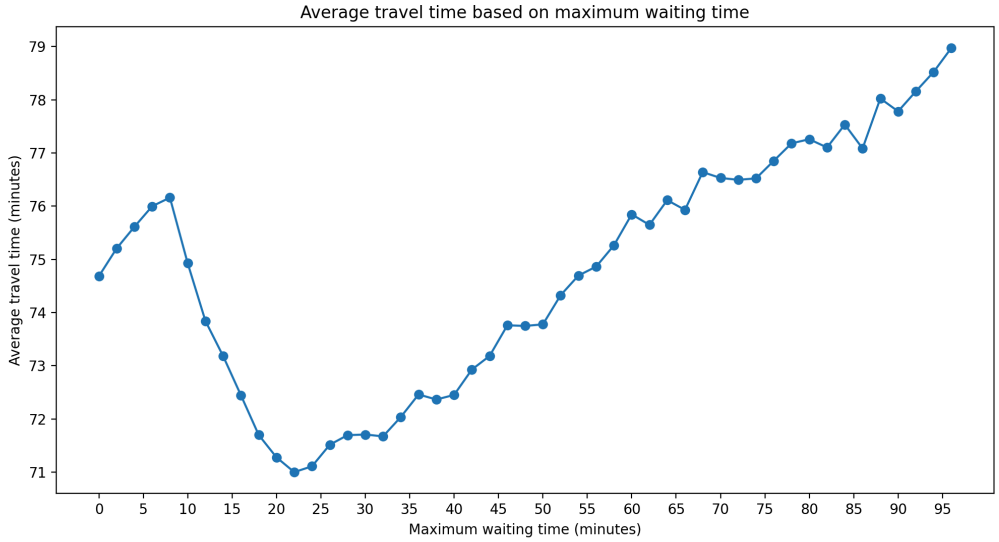

Figure 6: Experiment result showing the average travel time on a long track

5.2.4 Ratio of direct passengers to transfer passengers

This experiment evaluates the influence of how many direct passengers travel in the connecting train compared to transfer passengers. Intuitively, we would expect the simulation to return that it is more advantageous to wait on independent trains when the number of transfer passengers is greater. The track is the same as in the Short Track experiment.

The number of transfer passengers in the first simulation (more transfer passengers) are `Uniform(80, 100)`, `Uniform(50, 80)` and `Uniform(25, 50)` in each station, and the number of direct passengers are 15, 10, and 5 respectively.

The number of transfer passengers in the second simulation (less transfer passengers) are 70, 40 and 25 in each station, and the number of direct passengers are `Uniform(80, 100`, `Uniform(50, 80)` and `Uniform(25, 50)` respectively.

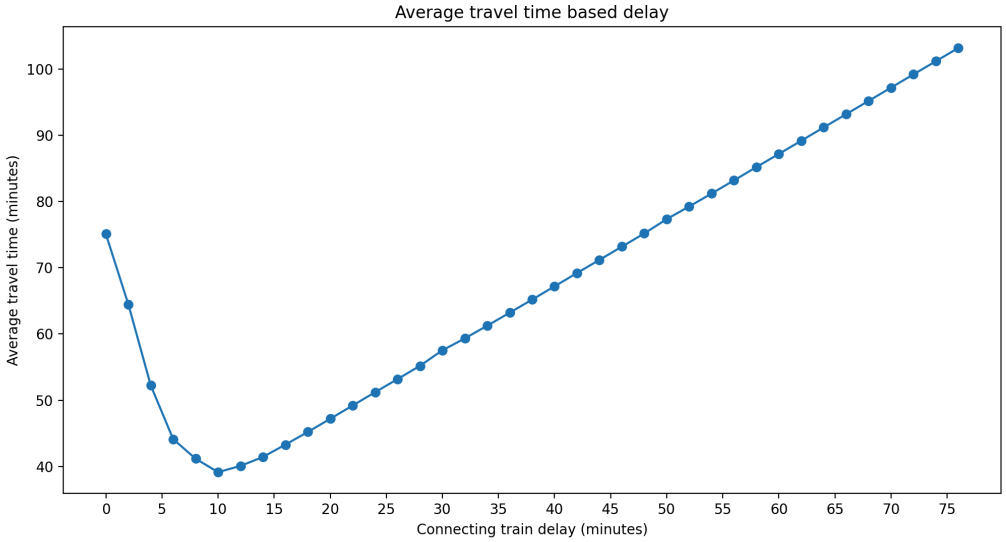

Figure 7: Experiment result with more transfer passengers

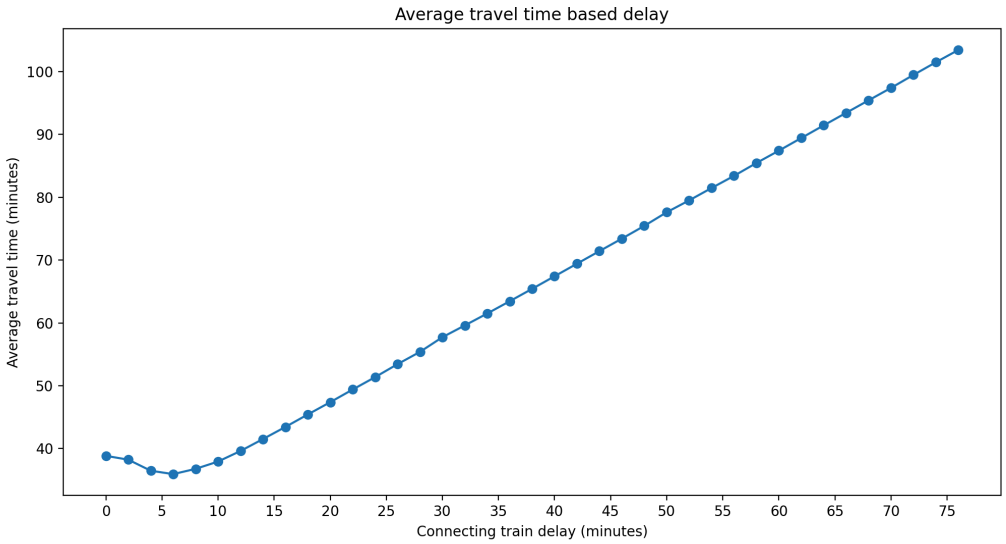

Figure 8: Experiment result with less transfer passengers

5.2.5 Real-world example

This experiment describes the real train S31 (Os 6040) traveling from Nymburk to Mladá Boleslav. It simulates the scenario of how advantageous it is to wait for the independent train S2 (Os 5822) traveling from Kolín to Nymburk. This independent train usually arrives with no delay (27 out of 32 instances of delays are up to 5 minutes, and 5 out of 32 delays are between 5 and 10 minutes). This is extracted from [6]. The experiment includes all stops [2], along with an estimated number of passengers boarding at each station based on the size of the station. The experiment shows that waiting for the independent train for approximately ten minutes is the most efficient policy overall, which aligns with the data from [6], indicating delayed departures of up to 15 minutes.

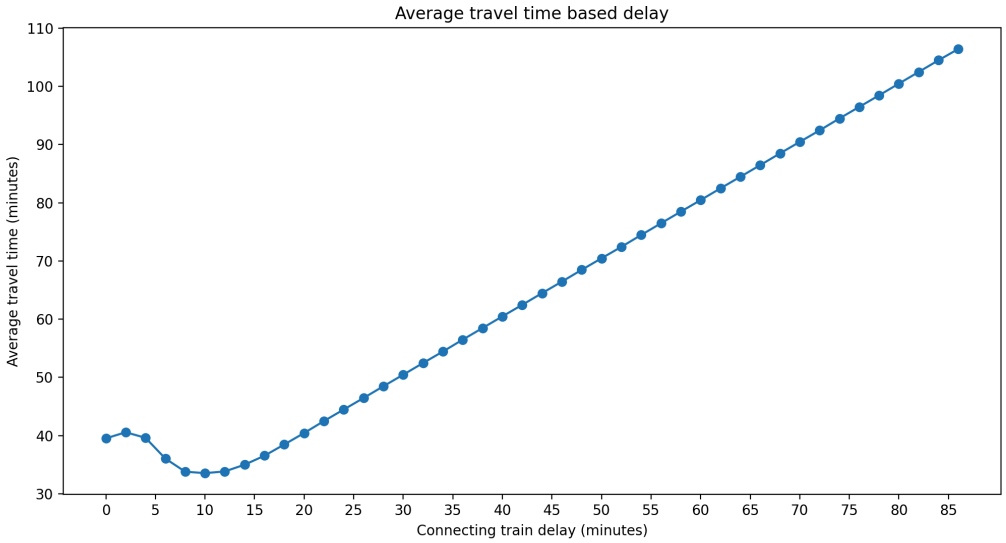

Figure 9: Experiment result showing the average travel time on a real-world track

5.2.6 Determining the optimal offset withing connecting train period

When experimenting with a modified version of our model, we created a simulation which had slightly different properties from our other simulations. Firstly, there were no direct passengers boarding in any station. Secondly, the maximum wait time was set to zero, therefore the connecting train did not wait for any other delayed trains. Thirdly, the connecting train was delayed by a fixed initial amount of time at the start of simulation. When we calculate the average travel time with different initial offsets, we are basically calculating the point within the interval of the connecting train period, at which it is best to schedule the connecting train for a given set of stations and independent trains. For example, given the connecting train period $T_c = 80\text{min}$, we are calculating, at which point in this 80 minute interval should the train be located at the initial (leftmost) station. This could be used to create new train schedules based on approximate numbers of transfer passengers and independent train delays.

The following result is from an example three-station track with an independent train carrying 50 passengers at each station. Note that the first average travel time (at offset of 0 minutes) is approximately the same as the last average travel time (at offset of 80 minutes). This is to be expected because the offset of 80 minutes is a multiple of the independent train period (80 minutes) and therefore the result should be the same as with no offset at all.

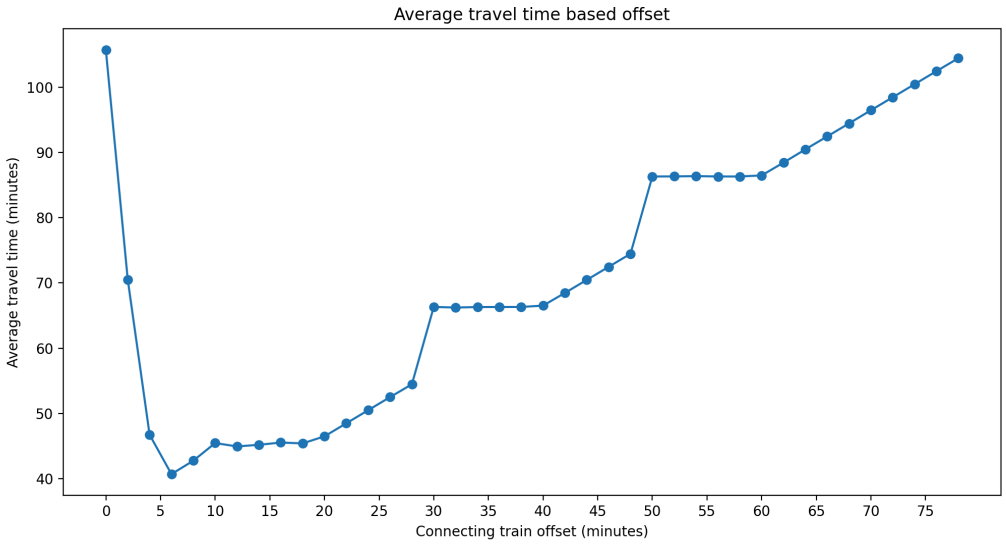

Figure 10: Experiment result of finding the optimal offset

5.3 Conclusion of Experiments

We ran experiments in which we spawned either just direct or just transfer passengers. In these experiments, we solved issues related to calculating train waiting times and passenger boarding. From the other experiments, we observed a few general facts about the system.

Firstly, there is a relationship between the ratio of transfer passengers and direct passengers on the optimal maximum wait time. The more transfer passengers there are compared to direct passengers, the more advantageous it is to allow for greater maximum wait time. At some point, there is no reason to wait for delayed trains at all since any created delays will impact the direct passengers. This threshold can be found using our simulation model simply by iterating through different values for the numbers of passengers and calculating the average travel time.

Secondly, the length of the main track impacts the optimal wait time. For short tracks where the connecting train returns to the same station after a relatively short time (in other words, the connecting train period is short), the optimal value is zero. This is because the transfer passengers do not have to wait long for the train to make a full cycle. Above a certain track length, it is advantageous to allow for some maximum wait time. Exactly how much can be calculated using our model.

6 Conclusion

In this work we created a model of a single railway track with a connecting train and an arbitrarily configurable number of stations and independent trains, for which the connecting train waits according to the policy used by the Czech Railways. We implemented the simulation model using SIMLIB and validated it against data collected on real-world railways. The model is implemented generically, and is therefore able to simulate a wide variety of specific real-world tracks simply by changing the track configuration and adjusting the stations' parameters. When simulating fictional tracks, the model produces results which satisfy basic reasoning. When the model's parameters are set to real-world data, the simulation results are aligned with data from the real track.

Bibliography

[1] "Jízdní řády IDS JMK." [Online]. Available: https://www.idsjmk.cz/timetables/links-routes

[2] "Traťové jízdní řády ČD." [Online]. Available: https://www.cd.cz/jizdni-rad/tratove-jizdni-rady/

[3] "Čekání na přípojné vlaky ČD." [Online]. Available: https://www.cd.cz/jizdni-rad/-25830/

[4] "Čekání na přípojné vlaky dopravce České dráhy." [Online]. Available: https://www.cd.cz/jizdni-rad/tratove-jizdni-rady/files/cz-znacky-171210-01.pdf

[5] "Koordinace tvorby jízdního řádu." [Online]. Available: https://www.cd.cz/jizdni-rad/-25517/

[6] R. Babilon, "Aktuální poloha vlaků ČD." [Online]. Available: https://kam.mff.cuni.cz/~babilon/zpmapa

[7] W. Hielscher, L. Urbszat, and C. Reinke, "On Modelling Train Traffic in a Model Train System." [Online]. Available: https://api.semanticscholar.org/CorpusID:53773396

[8] P. Peringer, "SIMLIB – Simulation Library for C++." [Online]. Available: https://www.fit.vut.cz/person/peringer/public/SIMLIB/

[9] P. Peringer, "Modelování a simulace." [Online]. Available: https://www.fit.vut.cz/person/peringer/public/IMS/prednasky/IMS.pdf