Towards Optimal Train Routing Using Microscopic Simulation on Moving Block Controlled Networks

Severin Lochschmidt, Stefan Engels* and Robert Wille
Chair for Design Automation, Technical University of Munich (TUM), 80333 Munich, Germany

Email: {severin.lochschmidt, stefan.engels, robert.wille}@tum.de

***Abstract*—The demand for sustainable railway transportation is increasing over time. At the same time, the capacity of railway networks is limited. Hence, efficient algorithms for generating optimal timetables are of great interest. Previous research focuses on trains being separated by classical fixed block signaling systems. With modern control systems based on moving block, e.g., within the *European Train Control System* (ETCS), the principles of safely separating trains change significantly. Only limited research on optimal routing on such modern railway networks exists. With this work, we propose a simulation approach tailored to be used with heuristic optimization algorithms to tackle this problem. Moreover, we show how such a framework can allow for more general inputs to jointly optimize what is usually planned sequentially as of today. The simulation framework is included within the open-source *Munich Train Control Toolkit* (MTCT) available on GitHub at https://github.com/cda-tum/mtct.**

I. Introduction

PASSENGER rail traffic in the European Union (EU) has increased by 35% in the last 30 years [1]. The main driver of this increase is high-speed rail traffic, which has risen on average by 6% per year. At the same time, freight rail traffic volume has stagnated, but, triggered by the European Green Deal, freight traffic should be prioritized to be transported by rail instead of road [2]. However, many railway lines are already operating at their capacity limit. A good planning process and traffic management systems are crucial to cope with the increasing demand. Some of the arising design tasks are introduced in [3].

The general railway planning process is commonly split into multiple sub-problems. Solving these sub-problems sequentially makes it possible to reduce complexity drastically. While mathematical modeling of these singular steps is feasible and widely implemented, an integrated approach remains out of reach [4], [5].

In this work, we focus on timetabling on a predefined railway network, or more precisely, how to optimally route trains through a railway network to fulfill certain conditions. The feasibility of a solution highly depends on the implemented control system. Since trains cannot operate on sight due to long braking distances, such signaling systems are crucial to prevent collisions and ensure safe operation. Classically, block signaling systems have been implemented. However, new train control principles have also been defined considering the increasing demand. One of these extensions is the introduction of *Moving Block* control systems, replacing classical block signaling with dynamic supervision. Relevant details on these systems are reviewed in Section II.

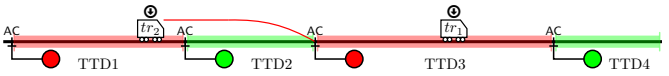
Much of the existing literature for train routing relies on the properties of fixed-block signaling. There is only limited research on optimally routing trains through networks equipped with modern moving block control systems [6], [7], [8]. They are based on Mixed Integer Linear Programming and do not (yet) scale well. We aim to close this gap by introducing an alternative heuristic approach based on simulation on a microscopic level, i.e., considering individual tracks. Existing simulation tools, such as OpenTrack [9] or OSRD [10], mainly focus on verifying and evaluating given solutions. Instead, our work focuses on applying a simulation framework already within the optimization process. Its architecture is built from the ground up with a moving-block approach, ditching old constraints. This allows us to apply novel techniques to encode solutions more efficiently to improve the performance of respective search algorithms applied to our simulation framework. Our approach can even include more complicated objectives. By doing so, it could be extended to combine multiple planning steps jointly in one optimization step. All work is integrated into the *Munich Train Control Toolkit* (MTCM) available open-source on GitHub at https://github.com/cda-tum/mtct.

For this Section II reviews the basic principles of train control systems, Section III describes the routing problem, Section IV introduces the underlying model and encoding with the aim of reducing the search space, Section V provides information on how this model can be used for optimization purposes, and, Section VI evaluates the approach by conducting a case study on a small benchmark set. Finally, Section VII concludes this work.

II. Train Control Principles

Train signaling has been based on splitting a line into discrete segments or blocks (fixed block), going back to the 19th century, Once a train enters a block, it is marked occupied, and access to other trains is forbidden. The status of a line segment is indicated to the driver via a mechanical display, called a semaphore, or later, an optical signal. This ensures mutual exclusion for track segments and thus prevents train collisions. *Trackside Train Detection* (TTD) uses electrical

*Corresponding author.



(a) Classical Block Signaling

(b) Moving Block Signaling

Fig. 1: Schematic drawings of signaling principles [8]

circuits embedded in the tracks or discrete sensors triggered by passing train axles to define the signal status.

Example II.1 (from [8]). *Consider two trains following each other on a single track as depicted in Figure 1a. Train tr_2 can only move until the end of $TTD2$. It cannot enter $TTD3$ because it is still occupied and, hence, might have to slow down in order to be able to come to a full stop before entering the occupied block section.*

With advances in communications technology, it has become possible to soften the traditional block structure into virtual and moving blocks. *Virtual Subsections* (VSS) obey the same principles as fixed blocks but can be arbitrarily defined. Position reporting relies no longer on trackside equipment but continuous bidirectional communication with the train. This alleviates the previously mentioned problems, especially with small or large block sizes, and allows for changing the block layout at any time.

Moving Block is a further evolution where blocks are abolished altogether. All trains in the network self-report and receive each other's position and speed. This allows them to calculate safe headway times dynamically, considering their individual braking curves. Moving Block has been taken up first on suburban railways, where risks are easier to manage. In metro lines, technical efficiency was improved by 11.5% by introducing moving block signaling and connected automation [11].

Example II.2 (from [8]). *In contrast to Eample. II.1, consider a moving block control implemented in Figure 1b. Because trains operate at the ideal absolute braking distance, tr_2 can move up to the actual end of tr_1 (minus a little buffer). In particular, it can already enter what has been $TTD3$ previously. Hence, trains can follow each other more closely.*

III. Train Routing under Moving Block Control

This work considers optimal train routing on a network equipped with a moving block control system. The *Train Timetabling Problem* (TTP) is part of the previously described planning process in Section I. It aims to find a feasible schedule and train routing under various constraints imposed by requirements on the desired timetable and safety conditions imposed by the control system as discussed in Section II. Train routing on a microscopic level consists of assigning trajectories and timings at the track level without imposing conflicts between different train movements.

In this work, we consider a part of a network that is equipped with a moving block control system, i.e., the problem is given as:

Problem III.1 (Train Routing under Moving Block Control)**.** ***Given:*** *A railway network, a list of trains including relevant properties, and a set of demands for every train consisting of*

- *information on when and where the train enters the network,*
- *information on when and where the train exits the network, and*
- *a list of stations*[1] *the train should stop at.*

Problem: *Find a feasible train routing that minimizes overall travel time.*

IV. Simulation Model

This section describes the underlying model of our microscopic time-continuous simulator. First, we briefly discuss how the railway network and train in Sections IV-A and IV-B. Finally, an efficient solution encoding is discussed with the aim of preventing invalid solutions from being generated by the optimizers presented in Section V.

A. Network and Train Model

Our network model is based on [3]. At the same time, we allow trains to move on railway tracks in arbitrary directions. Thus, a railway network is an undirected graph $G = (V, E)$ with vertices $V = \{v_1, v_2, v_3, \ldots, v_m\}$ representing railway switches, signals or other points of interest and edges $E = \{e_1, e_2, e_3, \ldots, e_m\}$, $e_i = \{v_j, v_k\} \subset V$ representing railway tracks. Each edge e_i also has an associated positive weight l_{e_i} representing track length and a maximum permitted speed $v_{e_i}^{max,edge}$. Moreover, railroad switches can only be traversed in specific directions, which is also modeled within the railway network.

B. Train Model

Trains are objects of a certain length l^{train}, maximal velocity $v^{max,train}$, and maximal acceleration/deceleration[2] $a^{max,train}$. We allow negative velocity values to include directional information. If $v > 0$, the train moves forward; if $v < 0$, it moves backward. In contrast to previous work [6], [7], [8], this allows trains to also turn around if necessary, e.g., in stations where this commonly happens in practice. Without loss of generality, we define a train's position as its center point and denote by $e[t] \in E$ the edge at time t

C. Solution Encoding

Choosing a purely heuristic approach affords us freedom in the design of the entire optimization loop. This makes it possible to limit the search space by intelligently preprocessing our candidate solutions. We choose a representation

[1]This only includes the station. The platform is not yet fixed, but it is to be decided by the solving algorithm.

[2]For simplicity, the train's maximal acceleration and deceleration are equal. However, the model can also be extended to model that these are usually different since braking and accelerating behavior differ.

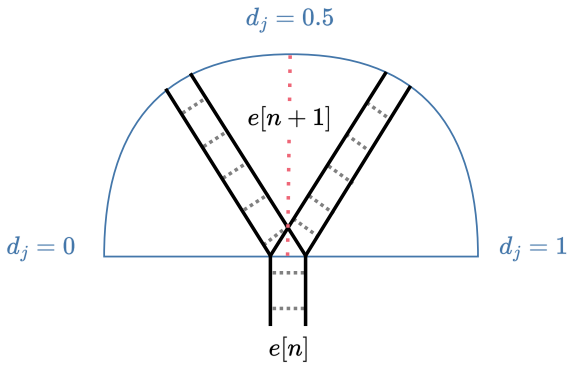


Fig. 2: Directional choice variable.

such that for each train, the decision variables map onto a physically possible train trajectory. This way, we eliminate solutions that contain overspeed, overacceleration, and invalid paths.

1) Railway Switches: Since trains in our model can reverse directions at any time, their trajectory describes an open, undirected walk. Instead of generating a valid graph walk for each solution, we compute it implicitly at simulation time with no additional cost. For this, we introduce directional choices $D = \{d_1, \ldots, d_{N^{max}_{transitions}}\}$, where each $d_j \in [0, 1]$ mimics a steering wheel for a given decision point. If $d_j = 0$, the train wants to move left; if $d_j = 1$, the train wants to move right; see Figure 2. Because $d_j \in \mathbb{R}$, this allows us to also choose middle tracks if a given switch allows for more than two options. At the first switch, the route is chosen according to d_1, at the second switch according to d_2, and so on. That means the route choice variables do not depend on time but are a list of decisions that are made in order whenever it is necessary to make a decision.

More precisely, assume that at time t, a train traverses a switch v_{trav} on edge $e[t]$ and has to make its kth decision. Assume that the list of possible successor edges allowed by the switch $\Gamma(e[t], v_{trav})$ is ordered and contains j elements $e_0, e_1, \ldots, e_{j-1}$. Then

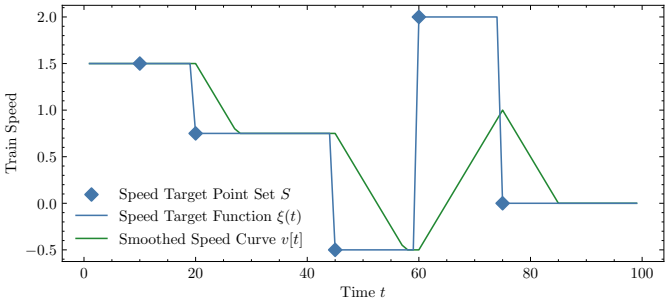
$$e[t + \Delta t] = e_{\lfloor (d_k \cdot (j-1)) \rceil}, \tag{1}$$

where $\lfloor \cdot \rceil$ denotes rounding to the closest integer.

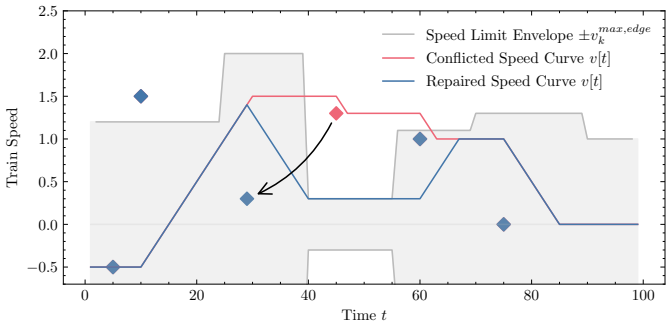
2) Speed and Acceleration: We do not directly encode each train's velocity (and acceleration) because modeling future speed limits (which might induce braking well in advance) would be challenging to model. Instead, we choose a set of speed target points S as decision variables, as time-velocity pairings. More precisely, any $s_i = (\tau_i, \nu_i) \in S$ consists of a time point $\tau_i \in [0, T_{max}]$ and a normalized velocity target $\nu_i \in [0, 1]$. From S we create a piecewise constant function $\xi(t)$ by previous-neighbor interpolation and scaling with $v^{max,train}$, i.e.,

$$\xi(t) = v^{max,train} \cdot \nu_{previous(S,t)}, \tag{2}$$

where $previous(S, t) := \max\{i \colon \tau_i \leq t\}$ is the most current speed target. From this, we induce the acceleration by attempting to reach the current speed target as quickly as possible, see Figure 3a, similar to a bang-bang controller known from optimal control theory. If $\left|\frac{\xi(t) - v[t]}{\Delta t}\right| \leq a^{max,train}$, then $v[t + \Delta t] = \xi(t)$, otherwise the train maximally accelerates toward its target.



(a) Transformation of speed decision variables into continuous curve.



(b) Single iteration of the track speed limit repair algorithm.

Fig. 3: Smoothing ξ to a feasible trajectory.

The track speed limit is much more difficult to encode since train speed $v[t]$ and edge position $e[t]$ are interdependent. Adjusting the speed can shift the time of an edge change or, when reversing, even the edges themselves. We leverage this by repairing $v[t]$ during the simulation using backtracking, a standard method for algorithmic constraint satisfaction as shown in Figure 3b. When encountering a constraint violation, the offending speed targets in S are removed and replaced with the maximum allowed speed for the track section. The simulation travels backward in time and applies just enough braking to avoid overspeed. Similarly, we can force the train to stop at a given point, e.g., at a station.

Encoding the target velocity curve as a set of discrete points in this way reduces the dimensions of the solution space. Any arbitrary curve in the solution space is still representable by increasing the number of points until every timestep is contained in S, amounting to a lookup table.

V. Optimization using Simulation

The combined encoding from Section IV-C gives us a space of decision variables consisting of speed target points S and direction choices D that map onto individually feasible train trajectories as previously discussed. Next, we evaluate the

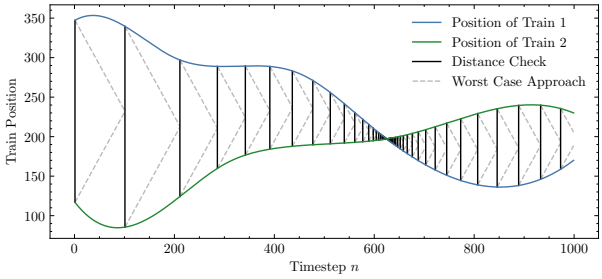


Fig. 4: Detection algorithm for headway constraint violations.

quality of each solution to define optimization criteria and apply various heuristic algorithms to find reasonable solutions.

A. Objective Function

The objective function is a composite of soft constraints $f_{headway}, f_{dest}$, and a global objective f_{stops}, which are explained next. They form the multi-objective minimization problem. Each objective can easily be normalized and weighted as desired.

1) Headway: Arguably, the most critical constraint is to avoid collisions, i.e., to apply long enough headways between two consecutive trains. It is introduced into the objective as a soft constraint $f_{headway}$ penalizing closeness beyond a minimum safety distance $h_{safe} \geq 0$.

Let $d_{i,j}(t)$ be the distance between tr_i and tr_jat time t. Additionally, a distance penalty function ω is chosen, which is zero for distances greater than h_{safe}, for example, a triangle function. The headway penalty can then be defined as

$$f_{headway} = \sum_{i=1}^{N_{trains}} \sum_{j=i+1}^{N_{trains}} \sum_{t} \omega(d_{i,j}(t)). \tag{3}$$

Using pre-computed all-pairs shortest paths for the entire network, the effort to find the distance between two trains is constant. The total number of necessary checks can be reduced by skipping time steps where a collision is physically impossible. For this, not that $d_{i,j}$ fulfills the following constraints:

$$\begin{aligned} & d_{i,j}(t) \geq 0, \\ & d_{i,j}(t) \leq \mathit{longest\ distance\ in\ network}, \\ & |d'_{i,j}(t)| \leq \left|v_i^{max,train}\right| + \left|v_j^{max,train}\right|, \text{ and} \\ & |d''_{i,j}(t)| \leq \left|a_i^{max,train}\right| + \left|a_j^{max,train}\right|. \end{aligned} \tag{4}$$

Using Equation (4), we can calculate a grace period for each distance check in which the trains cannot meet even when taking the shortest path towards each other at full speed. As seen in Figure 4, more checks are performed when trains are closer. Consequently, the computational effort for this objective varies based on the density and maximum speed of trains on the network. This method can be further improved by considering acceleration constraints in the grace period calculation and using a more efficient sampling of the distance function.

2) Destination: The problem specifies that each train enters and exits the network under consideration at some point in time with respective target velocities. The former constraint can be fulfilled by fixing the initial state of each train. Guidance toward a final destination is added through another soft constraint, penalizing the distance from the desired position and the difference in speed.

3) Intermediate Stops: Stops en route can be introduced by rewarding time spent with zero speed on scheduled edges. However, we can increase the prevalence of solutions with valid stops by encoding them using the repair algorithm described in Section IV-C. Each train passing an edge with a stop will be forced to halt for a minimum amount of time. In the objective evaluation, the fulfilled stops can be counted at minimal cost.

B. Optimizer

As described in Section IV, we are confronted with a general, non-linear, constrained optimization problem. In addition, the objective is non-convex and non-smooth, and its analytic form is unknown. We already defined an unconstrained version via relaxation using penalty functions; see Section V-A. This no longer guarantees solution feasibility for the combined objective. However, feasibility can be verified by evaluating the penalty functions.

The formulation gives rise to applying various heuristic approaches to find a good solution, which are described in the following paragraphs.

1) Random Search: A trivial baseline for search can be achieved by randomly assigning decision variables and keeping the best result. Such a method can be improved by introducing local search steps in each iteration. , e.g., using the Luus-Jakola method, a simple gradient-free local search algorithm [12]. An initial solution is randomly perturbed in a hypersphere with radius r. If the new candidate solution is an improvement, it replaces the old one as a starting point. The radius r contracts with each iteration by a contraction factor $c_{contract}$.

2) Greedy Search: It can be beneficial to mimic the behavior of human timetable planners when generating candidate solutions, as proposed in [13]. We, therefore, place trains sequentially using a limited random search for each instance in sequence. This approach breaks down the routing into subproblems. The order of train placement in our implementation is randomized. Classic asynchronous routing usually schedules trains by priority. Again, such an algorithm could be improved using local search, leading to the *Greedy Randomized Adaptive Search Procedure* (GRASP).

3) Genetic Algorithm: The use of genetic algorithms for timetabling is widespread [14], [15]. We implemented a primitive version, where crossover amounts to randomly choosing a trajectory for each train from the two parent solutions. Again, the iterations can be combined with local search. The population is sorted by objective score. Individuals are transferred from an elite set of top performers to the next

TABLE I: Search method comparison parameters.

Local Search and random+local and GRASP	Start Radius	$sr_{initial}$	0.4
	Stop Radius	sr_{final}	0.001
	Contraction Coefficient	$c_{contract}$	0.99
Greedy Search and GRASP	Per Train Stall Time		10ms
Genetic Search	Population	$\|P\|$	1000
	Elite Fraction	$n_{elite}/\|P\|$	0.01
	Mutation Rate		0.1
	Crossover Fraction	$n_{crossover}/\|P\|$	0.7

generation without modification. The crossover fraction determines the subset of individuals selected as parents for the next generation. Each offspring has a uniform chance to mutate, called the mutation rate; in this case, a random perturbation of all variables proportional to their maximum range is applied. The size of this perturbation is scaled down non-linearly as the generations progress. Our implementation uses the *openGA* library [16].

VI. Case Study

Due to their different model assumptions and significant limitations on publicly available data, the performance of timetabling tools is difficult to compare quantitatively. Ultimately, the most decisive measure is usefulness when applied in day-to-day planning. With this case study, we demonstrate the applicability of the proposed framework to the early stages of timetable planning.

A. Benchmark

For this, we solve three sample problems of varying size from [17]. These networks are included in the aforementioned *Munich Train Control Toolkit* (MTCT) available open-source. The first network, "Overtake," models a situation where faster trains must overtake slower ones. The second "SimpleNetwork" makes four trains cross at a central station. The last network, "Stammstrecke," is a full-sized replica of the Munich S-Bahn's core network passing through the city center.

Each train is defined with a fixed schedule for arrival and departure in the network and planned stops. In our test case, all objective functions are designed such that an objective score of zero indicates an optimal solution. This means all trains obey the distance minima, visit their scheduled stops, and reach their destination at the right speed and time. The collision objective is weighted roughly 10 to 1 against the two others.

Table I shows the parameters chosen to compare all search methods. They are a compromise across all test networks drawn from previous parameter measurements not reported in this paper due to space limitations. The numeric results for the search parameters we obtained are not universal since they depend heavily on problem parameters. However, we attempt to make generalized statements about algorithm performance wherever conclusive measurements allow.

B. Setup

Our C++ implementation is integrated into the open-source Munich Train Control Toolkit available on GitHub at https://github.com/cda-tum/mtct. Each optimizer instance was run on a single 4GHz core of a stock Intel i7-6700K processor using DDR4 RAM at 2133 MHz. For each algorithm instance, we record each improvement in objective score along with the expired search time. After collecting a statistically significant number of runs, we interpolate at 500 evenly spaced points over the runtime of the longest run. We then calculate the arithmetic mean score at each point. These mean score progressions can then be compared to assess algorithm quality.

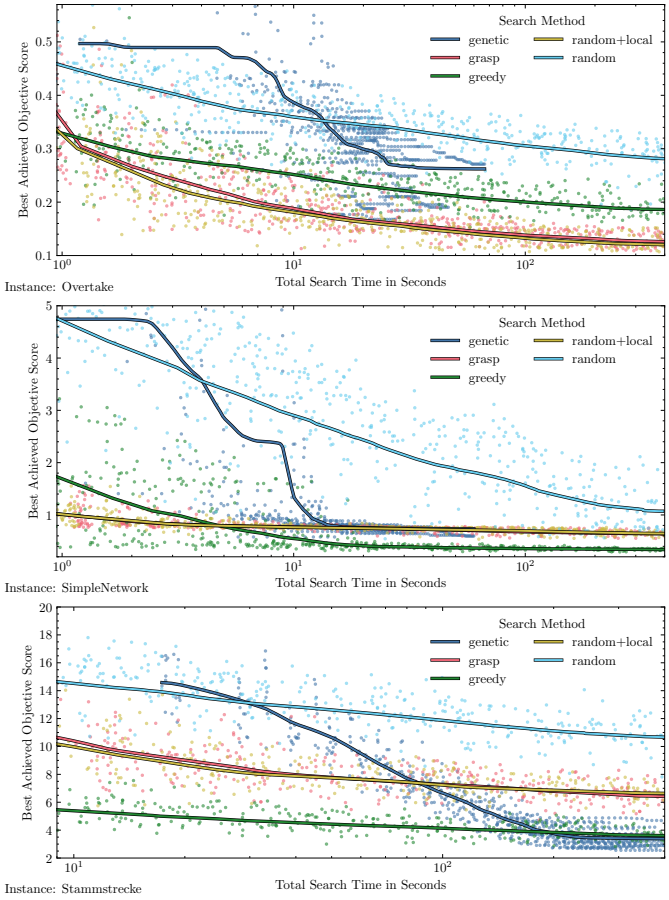


Fig. 5: Search Methods Effectiveness Measurement.

C. Evaluation

We compare the optimizers introduced in Section V-B. For the genetic algorithm, adding a local search step for each generation did not improve outcomes while increasing search time by an order of magnitude. For this reason, we excluded local improvement from the results. For our sample problems, all search methods stopped improving within the investigated time period. Thus, we focus on the best score before stalling, primarily reflecting solution quality.

Figure 5 pits all optimizers against each other. The random search baseline performed worst in all cases, as expected. The two local search methods dominated for the smallest instance. Focusing on refinement is more fitting for a smaller fraction of infeasible solutions. Greedy search was the overall best,

delivering both fast results and a low floor over time. It can efficiently generate many feasible solutions but will drop them immediately. The genetic algorithm showed progressively improved results for larger problem sizes, even overtaking greedy search in final solution quality for the Stammstrecke instance.

By tracking individual objectives and visualizing the train trajectories, we can assess the quality of the solutions for these candidates. The objective values are normalized to $[0, 1]$ or $[0\%, 100\%]$ respectively. The best solutions for the SimpleNetwork and Overtake instances exhibit no distance violations, an average destination penalty of 7% and 11%, and visit all reachable stops. The Stammstrecke solution does contain collisions, a 34% average distance penalty, and 27% average unfulfilled stops.

VII. Conclusions

We have reviewed current developments in railway planning and shown the need for modern, flexible, and accessible simulation platforms. We presented a new microscopic simulation framework for train timetabling on networks equipped with modern signaling systems based on moving block separation. This allows for more flexible train movements than previous approaches. We explained how architectural decisions and constraint encoding could be chosen to help heuristic algorithms find feasible trajectories quickly by implementing repair heuristics. Our approach allows for flexible objectives so that, e.g., line planning could be included in a joint optimization using the presented framework.

We tested multiple optimizer configurations on example timetabling problems to demonstrate functionality and gain insights into problem structure. Greedy and population-based search strategies proved superior to neighborhood-focused ones. For small instances, this trend was reversed. Overall, diversification, in combination with constraint satisfaction, dictated solution quality rather than local refinement. The smaller routing instances could be solved within a few minutes and withstand hard constraints even with limited hardware.

At the same time, this hints that the repair step introduced in the encoding to enforce feasible trajectories significantly helps the heuristics progress. Note that not all solutions firmly adhered to the headway constraints. Hence, the approach might improve from an enhanced solution encoding, which ensures safe distances by a backtracking repair step similar to the one proposed to ensure satisfying track maximum speeds. It is conjectured that this will highly benefit the quality of the solution.

Overall, this work constitutes a promising approach for using simulation frameworks to generate optimal timetables and microscopic routes on general railway networks equipped with modern moving block control systems. All code is available open-source within the *Munich Train Control Toolkit* (MTCT) on GitHub at https://github.com/cda-tum/mtct, which is still under active development.

References

[1] European Commission and Directorate-General for Mobility and Transport, *Transport in the European Union: Current trends and issues.* Publications Office of the European Union, 2024. [Online]. Available: https://doi.org/10.2832/131741

[2] European Parliament, "Position of the European Parliament and of the Council on the use of railway infrastructure capacity in the single European railway area," 2024, EP-PE_TC1-COD(2023)0271. [Online]. Available: https://www.europarl.europa.eu/doceo/document/TC1-COD-2023-0271_EN.pdf

[3] S. Engels, T. Peham, J. Przigoda, N. Przigoda, and R. Wille, "Design tasks and their complexity for the European Train Control System with hybrid train detection," *EURO Journal on Transportation and Logistics*, vol. 14, p. 100161, 2025. [Online]. Available: https://doi.org/10.1016/j.ejtl.2025.100161

[4] R. M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan, "Railway track allocation: models and methods," *OR Spectrum*, vol. 33, no. 4, pp. 843–883, dec 2009. [Online]. Available: https://doi.org/10.1007/s00291-009-0189-0

[5] N. Absi, C. Archetti, S. Dauzère-Pérès, and D. Feillet, "A two-phase iterative heuristic approach for the production routing problem," *Transportation Science*, vol. 49, no. 4, pp. 784–795, 2015. [Online]. Available: https://doi.org/10.1287/trsc.2014.0523

[6] T. Schlechte, R. Borndörfer, J. Denißen, S. Heller, T. Klug, M. Küpper, N. Lindner, M. Reuther, A. Söhlke, and W. Steadman, "Timetable optimization for a moving block system," *Journal of Rail Transport Planning & Management*, vol. 22, p. 100315, 2022. [Online]. Available: https://doi.org/10.1016/j.jrtpm.2022.100315

[7] T. Klug, M. Reuther, and T. Schlechte, "Does laziness pay off? - a lazy-constraint approach to timetabling," in *22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2022)*, vol. 106. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. [Online]. Available: https://doi.org/10.4230/OASIcs.ATMOS.2022.11

[8] S. Engels and R. Wille, "Comparing lazy constraint selection strategies in train routing with moving block control," in *Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS)*, vol. 39. Polish Information Processing Society, 2024, pp. 585–590. [Online]. Available: https://doi.org/10.15439/2024f3041

[9] Hürlimann, Daniel, "Objektorientierte Modellierung von Infrastrukturelementen und Betriebsvorgängen im Eisenbahnwesen," Ph.D. dissertation, 2002. [Online]. Available: https://doi.org/10.3929/ETHZ-A-004223151

[10] OSRD contributors. Open source railway designer. [Online]. Available: https://osrd.fr

[11] S. Canavan, D. J. Graham, P. C. Melo, R. J. Anderson, A. S. Barron, and J. M. Cohen, "Impacts of moving-block signaling on technical efficiency: Application of propensity score matching on urban metro rail systems," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2534, no. 1, pp. 68–74, 2015. [Online]. Available: https://doi.org/10.3141/2534-0

[12] R. Luus, *Direct Search Luus–Jaakola Optimization Procedure*. Springer Nature Switzerland, 2024, pp. 1–5. [Online]. Available: https://doi.org/10.1007/978-3-030-54621-2_129-1

[13] K. Nachtigall and S. Voget, "A genetic algorithm approach to periodic railway synchronization," *Computers & Operations Research*, vol. 23, no. 5, pp. 453–463, 1996. [Online]. Available: https://doi.org/10.1016/0305-0548(95)00032-1

[14] N. Bešinović, R. Goverde, G. Nicholson, and H. Steele, "Review of railway timetabling developments: planning phases, goals and future directions," 2021. [Online]. Available: https://www.researchgate.net/publication/353378414_Review_of_railway_timetabling_developments_planning_phases_goals_and_future_directions

[15] F. Hauck, "An overview and categorization of approaches for train timetable generation," 2023. [Online]. Available: https://doi.org/10.17169/REFUBIUM-37849

[16] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "OpenGA, a C++ genetic algorithm library," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2051–2056. [Online]. Available: https://doi.org/10.1109/smc.2017.8122921

[17] S. Engels, T. Peham, and R. Wille, "A symbolic design method for ETCS Hybrid Level 3 at different degrees of accuracy," in *23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023)*, vol. 115. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. [Online]. Available: https://doi.org/10.4230/OASICS.ATMOS.2023.6