Modeling Railway Systems in Answer Set Programming

Ryan Murphy[1]

[1]University of Potsdam, Germany

Abstract

Previous research has shown that developing train timetables for a single track is intractable. Modern railway systems, however, face growing complexities beyond just timetabling, for example in passenger demand, staffing requirements, rolling stock management, service disruptions, and energy consumption. The need for advanced solutions therefore has become inevitable. Answer set programming has demonstrated success on railway routing problems and has performed well in related problems such as multi-agent pathfinding, scheduling, and urban traffic management. The overall goal of this research is to provide a foundation for modeling large railway systems and their various challenges such that answer set programming can solve them in ways that are advantageous to the state-of-the-art.

1. Introduction

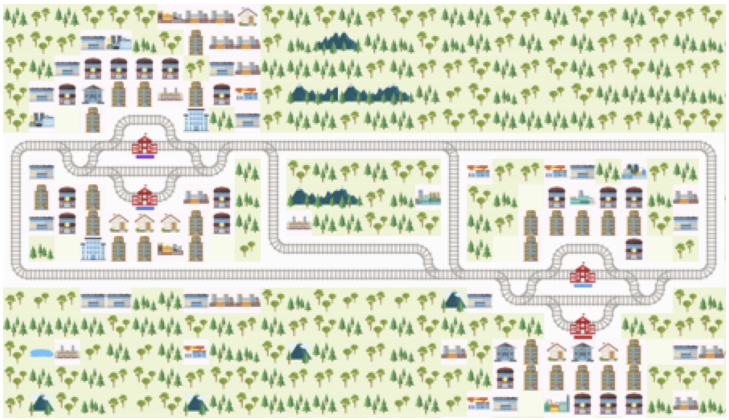


Figure 1: An example Flatland railway environment.

Railway systems face numerous operational challenges in planning and execution, and have been the subject of research for decades, particularly since the middle of the twentieth century. By 2002, it was proven that developing train timetables for a single track is NP-Hard [1]. As modern railway systems grow, both in terms of their networks and their passenger demand, the real problems they face become more complex and require more advanced solutions.

In 2019, tasked with solving such complex issues and inspired by advancements in digital technology, the Swiss Federal Railway (SBB), along with AIcrowd, founded *Flatland*, a global online railway scheduling and rescheduling competition to crowdsource novel approaches to modern-day problems [2]. An example Flatland environment is shown in Figure 1. My research began in 2023 as an attempt to use answer set programming (ASP) as an alternative to the machine learning approaches. This culminated in a master's thesis that outlined the creation of a toolkit now known as *flaspland* that provides an interface between *Flatland* written in Python and ASP written in clingo. The work also established standard fact formats for representing railway environments and put forth simple baseline encodings for safely navigating trains.

Joint Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026) co-located with the Federated Logic Conference (FLoC 2026).

ryan.murphy@uni-potsdam.de (R. Murphy)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-03

CEUR-WS.org/Vol-4311/DC_short_08.pdf

The research continues today in the form of Ph.D. work under the supervision of Prof. Dr. Torsten Schaub, and includes a graduate-level railway scheduling course, where students explore solutions to open research topics in this area, and theses written by highly-motivated students with a particular affinity for research at the intersection between knowledge representation and public transportation. In this paper, I will: provide **background context** into the history of railway research, modern approaches, and the advantages of answer set programming; outline my **research goals** for the next few years; highlight the **current status** of this research; share some **preliminary results**; and bring attention to **milestones** that lie on the horizon.

2. Background

Research into railway operations gained significant traction in the middle of the twentieth century, when such papers as *A Model for the Optimal Programming of Railway Freight Train Movements* in 1956 [3] and *Timetables for a Suburban Rail Transit System* in 1969 [4] sought to formalize problems in the domain by appealing to fields such as management science and linear programming. Over the following decades, the railway operations literature grew as researchers explored a growing variety of solutions involving dynamic programming and branch and bound procedures [5], integer programming and heuristics [6], and Lagrangian relaxation [7], among others.

In recent years, large railway operators have kept an open mind to novel approaches in machine learning, artificial intelligence, and other research fields as potential solutions to a growing number of problems in transportation. In 2019, the Swiss Federal Railway (SBB), along with AIcrowd, founded the Flatland Challenge [2], a global online railway scheduling and rescheduling competition in search of efficient ways of managing dense railway traffic. While the organizers have shown a particular interest in reinforcement learning and imitation learning solutions, they have attracted a wide array of approaches.

Answer set programming has already demonstrated success with problems in similar domains, such as multi-agent pathfinding [8], scheduling problems [9] [10], and traffic management [11]. Pathfinding is a key component of railway planning, and given that multiple trains share a common network with the mandate to find collision-free paths to their goals, one can view this as a multi-agent pathfinding problem. Similarly, since trains may be expected to follow a fixed timetable, one can view this as a scheduling problem, in which certain events must occur in a particular order or deadlines must be respected. Furthermore, the execution of railway activity may deviate from its intended plan, leading to congestion and delays. In this way, the central management of railway operations resembles the problem of urban traffic control.

In fact, answer set programming has even already been proven to work on railway-specific problems. The hybrid system *clingo*[DL] has been used to successfully create train schedules over a period of six hours for a portion of the network in Switzerland containing hundreds of train lines in a matter of minutes [12].

3. Research goals

The goal of this research is to establish best practices for modeling large railway systems in answer set programming. In short, this can be seen in three phases:

1. Formalization of railway environments
2. Solutions to the Flatland problem
3. Application to real challenges

3.1. Formalization

Our initial goal has been to formally understand large railway systems. By leveraging the Flatland ecosystem as a virtual railway generator and simulator, we have obtained an end-to-end view over a large

sample of environments. This means we have been able to consider a large diversity of characteristics in environments and iteratively test and refine our formal definitions. Additionally, developing a comprehensive formalization has enabled us to better relate the problems in modeling railway systems to those encountered in similar domains.

3.2. Solution

Following this, the aim is to exhaustively solve the Flatland problem, which necessitates a solution to both the vehicle scheduling problem (VSP) and the vehicle rescheduling problem (VRSP). Described in more detail in Section 6, the extension into VRSP requires the ability to handle stochastic malfunctions, which represent breakdowns and delays that occur every day in real railway networks. VSP instances take place during the operational planning phase and since these occur ahead of time, we can be more forgiving about how long it takes to find an acceptable solution. VRSP instances take place in response to unforeseen events, and since the surrounding conditions can change in real time, preference is given to finding a solution quickly over finding an optimal solution.

Furthermore, the quantitative evaluation of encodings gives us an idea of how ASP-based approaches compare to machine learning and other similar approaches. However, is not strictly necessary that the encodings outperform high-ranking encodings from the competition, as answer set programming provides important guarantees that other approaches do not, for example the avoidance of deadlocks and collisions, and that all trains reach their goals.

Finally, solving the Flatland problem requires accounting for trains that move at various speeds. The speed characteristic was initially included to simulate the discrepancies between fast-moving intercity trains and slow-moving local or freight trains. This concept has already been explored in our encodings, but is yet to be finalized.

3.3. Application

Beyond just Flatland, this body of understanding can be translated to critical problems faced by railway operators. Narayanaswami and Rangaraj classify problems in railway operations into broad categories from planning to execution [13]. Specific problems from across that spectrum that are worth addressing include

- routing and scheduling
- preemptive replanning for adverse events
- valid switch position logic (such as that used by the ILTIS system)
- energy utilization
- route synchronization
- modeling acceleration
- personnel management
- rolling stock management
- maintenance scheduling

Although solutions to many issues in the realm of transportation are rooted in public policy, it is my hope that this research can contribute to the solving of problems that require or can benefit from technological solutions.

4. Current status

We have submitted to ASPOCP 2026 a paper titled *Flaspland: A Test-bed for Routing and Scheduling in Answer Set Programming*. This paper provides a comprehensive formalization of the railway scheduling problem posed by Flatland, explores alternative formal representations of railway environments, and reviews experimental results on a variety of pathfinding encodings. For the following iteration of this paper, we have begun exploring encodings that leverage optimization and difference logic.

Beyond the most recent paper, I continue to maintain and improve the toolkit introduced in Secton 1 known as *Flaspland*. It serves as a bridge between the open-source Flatland code, which is written in Python, and answer set programming encodings written in clingo. There is a two-step asynchronous process. In the first step, an environment is generated from user-defined specifications and is converted into a clingo fact representation file. In the second step, the output of a pathfinding encoding (or set of encodings) is converted into a Python object that Flatland recognizes. Since the toolkit adheres to a common fact format, developers are free to explore various techniques and styles of encodings. Altogether, this allows for the seamless development, testing, debugging, and presentation of a wide variety of solutions.

In practice, the formalization outlined in the paper and the *Flaspland* toolkit are used by the students in our *Railway Scheduling* course at Universität Potsdam. We update the course each semester to provide them with the latest view of this research, and in turn the topics the students pursue in the course contribute to our broader understanding of modeling railway systems in answer set programming.

5. Preliminary results

One of the most interesting developments has been gaining a better understanding of how the Flatland problem differs from other well-explored domains, for instance multi-agent pathfinding. In common multi-agent pathfinding problems, agents exist in a grid and have a consistent number of degrees of freedom, meaning aside from the presence of obstacles or boundaries, agents are free to move to any unoccupied adjacent cell. In Flatland, movements become more restricted, for example because trains may only move forward, meaning we have to preserve information about their orientation. Furthermore, trains are bounded to the topology of the track, limiting their possible transitions to only one or two adjacent cells, shown in Figure 2.

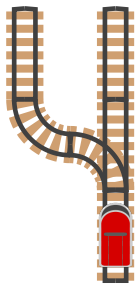


Figure 2: A small railway environment.

To take advantage of these more narrowly-defined conditions, the aforementioned *Flaspland* paper puts forth a hypothesis that alternative representations of the environment yield performance advantages over the standard grid-based environment as the problem scales. We first developed encodings that translate the environment into various graph structures, and then use a common pathfinding encoding, whose rules correspond to those defined in our formalization, to calculate collision-free paths along each of the graphs. To supply a baseline for comparison, we retrofitted an existing multi-agent pathfinding encoding [14] that takes as input the standard grid-based environment. The results show that as the number of agents in an environment grows, the alternative representations outperform the baseline [15]. While the alternative approaches tend to fall behind in grounding time, they make up for it in solving time. This suggests that there may yet be potential improvements if grounding can be further reduced. From these initial results, we are excited to explore different styles of encodings, further alternative representations of environments, and difference logic, to name a few.

6. Open issues and expected achievements

The next significant milestone has to do with handling stochastic malfunctions. So far, they have been overlooked. Generally, railway routing and scheduling problems can be viewed in two stages:

1. **Planning.** Planning under ideal conditions is known as the vehicle scheduling problem (VSP).
2. **Execution.** Developing new plans under realistic conditions once execution has already begun is known as the vehicle rescheduling problem (VRSP).

So far, our encodings have been well suited for the VSP, as we pass along plans in a single-shot solving fashion. In the future, we will need to adjust our architecture to function as a two-way channel, to pass

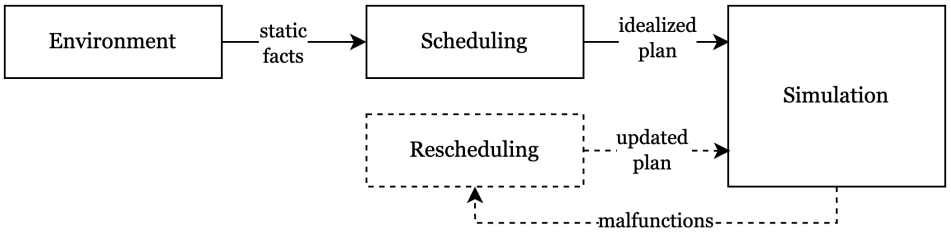


Figure 3: Logical flow of the *flaspland* system. The dashed lines represent the proposed extension to handle malfunctions

forward calculated train plans, as well as to report back changes from the simulation to our encodings, as shown in Figure 3. Concurrently, the encodings will need to shift toward a multi-shot solving paradigm, to account for changes in current conditions.

In principle, the logic of a VSP encoding could be recycled by treating a the state of the simulation following a malfunction as a *fresh start* (i.e. a new environment, as if the current state of the trains were their new starting conditions), but if conventional wisdom prevails, benefits exist by retaining information that has already been learned.

Beyond this, my sights are set on tackling the individual problems that comprise railway systems. We will continue to build a community of those who enroll in our railway scheduling graduate course and those who choose to write the theses with us on this topic.

7. Declaration on Generative AI

The author has not employed any generative AI tool in the preparation of this work.

References

[1] A. Caprara, M. Fischetti, P. Toth, Modeling and solving the train timetabling problem, Oper. Res. 50 (2002) 851–861.

[2] S. Mohanty, E. Nygren, F. Laurent, M. Schneider, C. Scheller, N. Bhattacharya, J. Watson, A. Egli, C. Eichenberger, C. Baumberger, G. Vienken, I. Sturm, G. Sartoretti, G. Spigler, Flatland-rl : Multi-agent reinforcement learning on trains, CoRR (2020). URL: https://arxiv.org/abs/2012.05893.

[3] A. Charnes, M. Miller, A model for the optimal programming of railway freight train movements, Management Science 3 (1956) 74–92.

[4] F. Salzborn, Timetables for a suburban rail transit system, Transportation Science 3 (1969) 297–316.

[5] E. Petersen, R. Merchant, Scheduling of trains in a linear railway system, INFOR: Information Systems and Operational Research 19 (1981) 246–258. URL: https://doi.org/10.1080/03155986.1981.11731826. doi:10.1080/03155986.1981.11731826. arXiv:https://doi.org/10.1080/03155986.1981.11731826.

[6] X. Cai, C. Goh, A fast heuristic for the train scheduling problem, Computers & Operations Research 21 (1994) 499–510. URL: https://www.sciencedirect.com/science/article/pii/030505489490099X. doi:https://doi.org/10.1016/0305-0548(94)90099-X.

[7] U. Brännlund, P. Lindberg, A. Nõu, J. Nilsson, Railway timetabling using lagrangian relaxation, Transp. Sci. 32 (1998) 358–369.

[8] V. Nguyen, P. Obermeier, T. Son, T. Schaub, W. Yeoh, Generalized target assignment and path finding using answer set programming, Proceedings of the International Symposium on Combinatorial Search 10 (2021) 194–195. URL: https://ojs.aaai.org/index.php/SOCS/article/view/18487. doi:10.1609/socs.v10i1.18487.

[9] M. Banbara, K. Inoue, B. Kaufmann, T. Okimoto, T. Schaub, T. Soh, N. Tamura, P. Wanko, teaspoon: Solving the curriculum-based course timetablingpproblems with answer set programming, Annals of Operations Research 275 (2019) 3–37. URL: https://doi.org/10.1007/s10479-018-2757-7. doi:10.1007/s10479-018-2757-7.

[10] M. El-Kholany, M. Gebser, K. Schekothin, Problem decomposition and multi-shot ASP solving for job-shop scheduling, tplp 22 (2022) 623–639. doi:10.1017/S1471068422000217.

[11] A. Tarzariol, , M. Maratea, M. Vallati, A casp-based solution for traffic signal optimisation, Theory and Practice of Logic Programming 25 (2025) 794–812. URL: http://dx.doi.org/10.1017/S1471068425100306. doi:10.1017/s1471068425100306.

[12] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti, P. Wanko, Train scheduling with hybrid ASP, tplp 21 (2021) 317–347. doi:10.1017/S1471068420000046.

[13] S. Narayanaswami, N. Rangaraj, Scheduling and rescheduling of railway operations: A review and expository analysis, Technol. Oper. Manag. 2 (2011) 102–122.

[14] J. Behrens, R. Kaminski, T. Schaub, T. Son, J. Švancara, P. Wanko, Routing and scheduling in answer set programming applied to multi-agent path finding: Preliminary report, CoRR abs/2403.12153 (2024). URL: https://arxiv.org/abs/2403.12153.

[15] R. Murphy, A. Nemes, J. Romero, T. Schaub, Flaspland: A testbed for routing and scheduling in answer set programming, Pending (2026).