Tackling Train Routing via Multi-agent Pathfinding and Constraint-based Scheduling

Jiří švancara[a] and Roman Barták[b]

Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic

Keywords: Train Routing, Multi-agent Pathfinding, Scheduling, Satisfiability, Constraint Satisfaction.

Abstract: The train routing problem deals with allocating railway tracks to trains so that the trains follow their timetables and there are no collisions among the trains (all safety rules are followed). This paper studies the train routing problem from the multi-agent pathfinding (MAPF) perspective, which proved very efficient for collision-free path planning of multiple agents in a shared environment. Specifically, we modify a reduction-based MAPF model to cover the peculiarities of the train routing problem (various train lengths, in particular), and we also propose a new constraint-based scheduling model with optional activities. We compare the two models both theoretically and empirically.

1 INTRODUCTION

The classical train timetabling problem involves constructing a timetable for (usually) passenger trains respecting various constraints such as capacity restrictions and service quality. The timetable is constructed once for a given period, and then the trains follow it in day-to-day service. However, additional trains may appear during the daily routine, typically freight trains with irregular service, and unexpected situations, such as track closures, accidents, and delays, may happen. A train dispatcher's role is to decide if the trains can move safely through a given area and allocate specific tracks to the trains. In this paper, we study the automation of this *train routing problem*. The problem is given by a track infrastructure (railway network) and a train timetable describing initial and target locations of trains with possible stops and expected departure/arrival times. The task is to allocate trains to tracks and times so that there are no collisions among the trains and the train timetable is followed.

The train routing problem has been addressed (partially) in Flatland Challenge (Mohanty et al., 2020) that has been proposed to answer the question "How to manage dense traffic on complex rail networks efficiently." Three large railway network operators, Swiss Federal Railways, Deutsche Bahn, and SNCF, are among the challenge's organizers, which highlights the importance of the problem. Though the challenge is specifically designed to foster progress in multi-agent reinforcement learning, the winning systems in recent rounds are based on multi-agent pathfinding technology (Li et al., 2021).

Unlike the Flatland Challenge, we assume trains of various lengths, which makes the problem more realistic. We focus less on the dynamics of the environment with immediate reaction to the current situation. We emphasize batch processing, where the tracks need to be allocated for a given set of trains for a given period. This problem is essential for train dispatchers as it provides them the information if (and how) a given set of trains can go through their area of control. As the MAPF technology proved successful for the Flatland challenge, we start with modifying the MAPF model to support trains of various lengths. We identify the reasons why a direct translation of train movement to MAPF is not possible. Motivated by a constraint-based scheduling approach to MAPF (Barták et al., 2018) that supports more general capacity constraints, we propose a constraint model with optional activities to describe the train routing problem fully. This declarative model has the advantage of adding other constraints, for example, expressing various safety measures such as train separation and using various objectives, such as following the train schedule as close as possible. The paper demonstrates that constraint-based scheduling with optional activities is a viable approach to address the train routing problem.

[a] https://orcid.org/0000-0002-6275-6773
[b] https://orcid.org/0000-0002-6717-8175

Švancara, J. and Barták, R.
Tackling Train Routing via Multi-agent Pathfinding and Constraint-based Scheduling.
DOI: 10.5220/0010869700003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 1*, pages 306-313
ISBN: 978-989-758-547-0; ISSN: 2184-433X

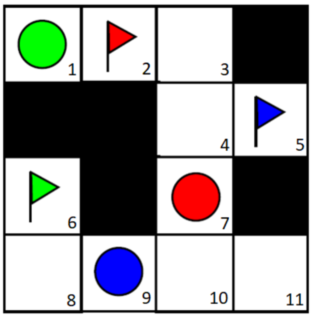


Figure 1: An example of MAPF instance. The white cells are vertices of a 4-connected grid graph, the colored circles denote initial locations of the agents, and the colored flags denote the goal locations of the agents.

Table 1: A solution to the MAPF instance defined in Figure 1. The numbers correspond to the vertices each agent occupies at that timestep.

Timestep	0	1	2	3	4	5	6	7	8	9
Agent Green	1	2	3	4	7	10	9	8	6	6
Agent Red	7	4	5	5	4	3	2	2	2	2
Agent Blue	9	10	11	11	11	11	10	7	4	5

2 BACKGROUND on MAPF

The *multi-agent pathfinding (MAPF) problem* is defined as a pair (G,A), where G is a graph $G = (V,E)$ and A is a set of agents. Each agent $a_i \in A$ is defined as a pair of vertices $a_i = (s_i, g_i)$ which represent its starting and goal vertex respectively. At each timestep, all of the agents perform an action synchronously. The action can be a move action to a neighboring vertex or a wait action (i.e., staying in the same vertex).

The task is to find a sequence of locations (alternatively, a sequence of actions) for each agent such that no two agents collide with each other. There are different types of collisions defined in the literature (Stern et al., 2019). The two most often used are vertex collision – two agents are present in the same vertex at the same time, and swapping collision – two agents are moving on the same edge in the opposite directions at the same time. Forbidding these two collisions ensures that no two agents are present at the exact physical location simultaneously. We call this parallel motion since it allows the agent to move closely one after another like a train, which will be helpful to us. An example of a MAPF instance can be seen in Figure 1, while a solution for that instance can be seen in Table 1. The table states the location for each agent at each timestep.

We do not provide a formal definition for a train movement problem since the specific details will arise from the formalism we use to solve the problem. However, the task is quite intuitive and close to the MAPF problem. We are given a shared environment represented by a graph and a set of trains. Each train is associated with a given initial position and the desired goal position. In our models, we further assume that each train may have a different length; therefore, the train may occupy a different number of vertices at a given time, each train is of length at least two, and the trains are not allowed to move backward. The task is to navigate each train to the desired destination without any collision with the other trains.

2.1 Modeling Large Agents

The models proposed in this paper are built as an extension of existing MAPF models. There exist extensions of the classical MAPF model to deal with large agents, but as we shall show now, these extensions are inappropriate to model trains of various lengths.

Generalization of MAPF for heterogeneous agents explores the setting where each agent occupies multiple vertices in a homogeneous grid map (Atzmon et al., 2020). With this approach, it is possible to model a long agent that resembles a train, and while the algorithm allows for the agent to turn, the agents are considered to be rigid (i.e., not changing their shape). This property may cause an instance to be unsolvable in cases where an agent that may change shape would be able to find a solution. Such example can be seen in Figure 2. The trains need to be able to bend around corners, thus changing the shape.

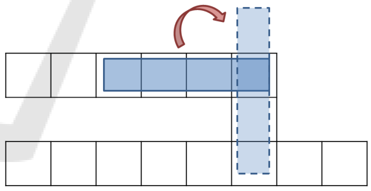


Figure 2: Example of an instance that is unsolvable for a rigid agent.

Another approach that deals with agents of varying size (Andreychuk et al., 2019), considers the agents to be circular. In the paper, the agents move continuously, and a collision is detected if any two agents overlap. It is possible to model a train by a circle of sufficient diameter. This representation is valid even on bends; however, two circular agents may collide in places where the physical trains are not present. See Figure 3 for an example.

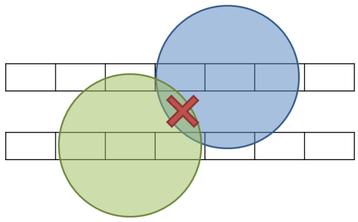


Figure 3: Example of an instance that is unsolvable by representing trains as large circular agents. The agents are moving in opposite directions and the circular agents cause a collision that would not happened in the physical world.

3 MODELING TRAIN MOVEMENT

The main difference and the hardness of the problem arise from the fact that a train may occupy more than one location at a given time. New constraints need to be set such that no two trains collide anywhere in these occupied locations. Furthermore, compared to the mentioned related work, the shape of the train may change based on the section of the track the train is using. There exists an approach called *Multi-Train Path Finding* (MTPF) (Atzmon et al., 2019) that is very close to this problem. However, in MTPF the train expands from a start node and shrinks to the destination node, while the trains preserve their length in reality. We formulate two different approaches to model the train movement.

3.1 MAPF Model

To model the trains' movement, we treat each train as a group of agents $T_1, \dots T_n$ such that $\bigcup_{T_1,\dots T_n} = A$. The number of agents in each group is determined by the length of a given train. The problem is then to navigate the group to the goal location in such a way that the agents not only remain connected but also maintain the structure of the train. More specifically, this means that when the group moves, each agent moves to the location that was occupied in the previous timestep by the agent next to it. Similarly, if the group waits (perform no-op) each agent in the group waits.

We start by using a MAPF solver based on a reduction to Boolean satisfiability (SAT). This solver models the classical MAPF problem as was defined in the previous chapter.

Let's assume that we are looking for a solution to the MAPF problem with makespan M using the parallel motion restriction on allowed conflicts. We define the following two sets of variables: $\forall v \in V, \forall a_i \in A, \forall t \in \{0,\dots,M\} : At(v,i,t)$ meaning that agent a_i is at vertex v at timestep t; and $\forall (u,v) \in E, \forall a_i \in A, \forall t \in \{0,\dots,M-1\} : Pass(u,v,i,t)$ meaning that agent a_i goes through an edge (u,v) at timestep t. More specifically, it starts traversing the edge at timestep t and enters the vertex v at timestep $t+1$. This is why the variables are not defined for timestep M. An auxiliary edge (v,v) is added to E, thus $Pass(v,v,i,t)$ means that agent a_i stays at vertex v at timestep t. To model the MAPF problem, we introduce the following constraints:

$$\forall a_i \in A : At(s_i,i,0) = 1 \tag{1}$$

$$\forall a_i \in A : At(g_i,i,M) = 1 \tag{2}$$

$$\forall a_i \in A, \forall t \in \{0,\dots,M\} : \sum_{v \in V} At(v,i,t) \le 1 \tag{3}$$

$$\forall v \in V, \forall t \in \{0,\dots,M\} : \sum_{a_i \in A} At(v,i,t) \le 1 \tag{4}$$

$$\forall u \in V, \forall a_i \in A, \forall t \in \{0,\dots,M-1\} : \\ At(u,i,t) \implies \sum_{(u,v) \in E} Pass(u,v,i,t) = 1 \tag{5}$$

$$\forall (u,v) \in E, \forall a_i \in A, \forall t \in \{0,\dots,M-1\} : \\ Pass(u,v,i,t) \implies At(v,i,t+1) \tag{6}$$

$$\forall (u,v) \in E : u \neq v, \forall t \in \{0,\dots,M-1\} : \\ \sum_{a_i \in A} (Pass(u,v,i,t) + Pass(v,u,i,t)) \le 1 \tag{7}$$

Constraints (1) and (2) ensure that the starting and goal positions are valid. Constraints (3) and (4) ensure that each agent occupies at most one vertex and every vertex is occupied by at most one agent. The correct movement in the graph is forced by constraints (5) – (7). In sequence, they ensure that if an agent is in a certain vertex, it needs to leave it by one of the outgoing edges (5). If an agent is using an edge, it needs to arrive at the corresponding vertex in the next timestep (6). Finally, (7) forbids two agents to traverse two opposite edges at the same time (forbidding swapping conflict). To find the optimal makespan, we iteratively increase M until a satisfiable formula is generated.

To express that the group of agents moves as a single train, we add the following constraints.

$$\forall j \in \{1,\dots n\}, r \in T_j, \forall a_i \in T_j, a_i \neq r, \\ \forall t \in \{0,\dots,T-1\} : \\ \sum_{v \in V} Pass(v,v,r,t) = \sum_{v \in V} Pass(v,v,i,t) \tag{8}$$

$$\forall j \in \{1,\dots n\}, \forall a_i \in T_j, a_i \neq a_l, \\ \forall t \in \{0,\dots,T-1\}, \forall v \in V : \\ At(v,i,t) = 1 \wedge At(v,i,t+1) = 0 \\ \implies At(v,i+1,t+1) = 1 \tag{9}$$

Assume that there are n groups of agents (trains). For each group, we can choose a representative r. Also to simplify the notation, r is the first agent in the group, and agents' indexes are ordered in the same way as the agents are ordered in the train representation. This means that agent a_l in a train of length l is the agent at the very end. The constraint (8) represents a situation when the train waits. It ensures that if one of the agents in that group is using a wait edge, all of the agents also use wait edges. The constraint (9) ensures that if an agent a_i moves to a new location (i.e., it is not staying in the same location v in two consecutive timesteps), the agent behind him, with the index of $+1$, moves to the previous location v of agent a_i. The constraint that trains are not allowed to move backward arises from constraint (9) and the fact that trains are of length at least two – the leading agent r of the train cannot go back as that vertex is occupied by the second agent of the train.

Since we are considering that trains are of length greater than one, we do not need to include the constraint (7) that forbids swapping of agents. Swapping of two trains of length at least two would lead to a vertex collision.

We provide the constraints as a set of inequalities rather than a CNF formula since it is more readable and there are tools that automatically translate such inequalities into a CNF that is solvable by any SAT-solver (Barták et al., 2017).

Another idea of how one might try to simulate the train motion is to use just one agent to represent a train and use k-robust MAPF (Atzmon et al., 2017). In this version of MAPF, it is required for each vertex to be empty for at least k timesteps before another agent may enter it. If k is set as the length of the longest train, the k vertices behind each agent, where the rest of the train should be present, are forced to be empty, thus producing a valid plan. However, a problem arises when the agent stops. Each timestep the agent remains stationary the number of unusable vertices behind the agent decreases by one. Eventually, if the agent does not move for k timesteps another agent may move right next to it, which would create a conflict in the real world. An example can be seen in Figure 4.

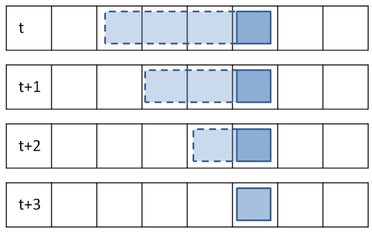


Figure 4: Example of 3-robust MAPF simulating train of length 4. The agent is not performing a move action in timesteps $t,\dots,t+3$, thus the length of the simulated train is "shrinking".

3.2 Scheduling Model

Our scheduling-based model is inspired by a scheduling approach for solving classical MAPF and an extension of MAPF where edges may be assigned lengths and capacities (Barták et al., 2018).

Similarly, we model the problem in the Constraint Programming formalism borrowing ideas from scheduling and routing problems. We see the possible locations as resources with limited capacity. We use the concept of optional (alternative) activities (Laborie et al., 2009) with specialized global constraints, such as *NoOverlap*, modeling resources.

In contrast to the MAPF model, in our scheduling model, we do not treat each vertex of the input graph as a single location. This granularity is unnecessarily fine. Considering that the railway network usually consists of long stretches of tracks with no intersections, we condense these long stretches into a single location, while intersections, where the train may change its direction to move to other rail segments, are kept with the high granularity. Figure 5 shows how a real railway network is converted to a set of activities for a specific train. Another example is given in the experimental section in Figure 9 showing how a MAPF-like model of the railway network is condensed in the scheduling model (vertices correspond to rail segments occupied by transport activities). Note that this high abstraction is not possible in the MAPF model, since condensing multiple vertices into a single one would lose the information on travel time. This is not an issue for the scheduling model, because we can assign different durations to activities modeling the train movement.

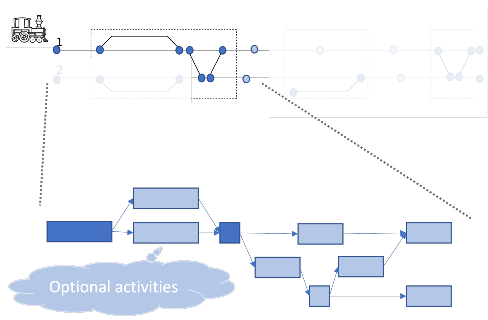


Figure 5: Converting a railway network to a set of transport activities.

Given a set of n trains $T_1,\dots,T_n$ and m locations $L_1,\dots,L_m$ created from the original graph G, we create an interval optional variable $Traverse(T_i, L_j, L_k)$ meaning that train T_i is traversing location L_j, coming from location L_k. It is needed to know the previous location of the train because this can determine what are the next possible locations and activities (recall that trains may not simply turn around or go backward), in other words, the previous location determines the orientation of the train in the current location. The activities are interval activities, meaning that they are assigned duration and may occur in some predefined time. See Figure 6 for visualization. The duration of the activity is determined by the length of the train plus the length of the rail segment and the speed of the train. The activity directly corresponds to the time any part of the train is present anywhere in the given rail segment.

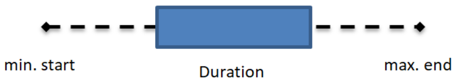


Figure 6: Visualization of an activity duration over the possible start and end times.

Given a start location L_{s_i} of train T_i, we write $Traverse(T_i, L_{s_i}, -)$ as the initial activity of train T_i. Similarly, we have the final activity $Traverse(T_i, L_{g_i}, L_k)$ for a goal location L_{g_i} of train T_i. In this case, the previous location is needed because in the problem specification we require the train to arrive in the goal station with specific orientation.

The initial and final activity of each train is required, we set the following constraints.

$$\forall i \in \{1, \dots, n\}: \quad PresenceOf(Traverse(T_i, L_{s_i}, -)) = 1 \tag{10}$$

$$\forall i \in \{1, \dots, n\}: \quad PresenceOf(Traverse(T_i, L_{g_i}, L_k)) = 1 \tag{11}$$

The presence of the other activities is determined by the solver based on the following constraints.

$$\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\}: \quad Alternative(Traverse(T_i, L_j, L_k), Next(T_i, L_j, L_k)) \tag{12}$$

$$\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\}: \quad Alternative(Traverse(T_i, L_j, L_k), Prev(T_i, L_j, L_k)) \tag{13}$$

The definition of the *Alternative* constraint, which gets an interval variable as the first argument and a set of interval variables as the second argument, is as follows. If the activity given as the first argument is present, then exactly one activity from the set of activities given as the second argument is present. In addition, it gives a constraint on the start and end time of the activities. The constraints (12) and (13) ensure that if an agent is traversing a location, it will next traverse one of the possible adjacent locations given the orientation of the train. Similarly, before traversing a location, the train needed to be traversing one of the locations that can lead it to the current one. The Possible transitions between activities need to be figured out as an input of the model. However, this is easily done by a simple traversal of the underlying graph.

The start and end times of the following activities need to be set. To ensure the correct movement of the trains with no swapping of the trains, we introduce a negative time delay between the two following activities. See Figure 7 for an illustration. In the first activity the train is traversing one rail segment, while in the other activity, the train is traversing some adjacent rail segment. During the t timesteps the train is performing both activities; the train is in fact occupying both rail segments at the same time. Therefore this time delay is determined by the length of the train and train speed. Basically, it defines the time that the train needs to pass a border point between two successive railway segments.

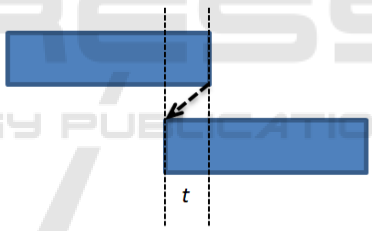


Figure 7: Visualization of two activities being performed after each other with the negative time delay t.

The negative time delay is enough to enforce no swapping among the trains. To forbid the trains to occupy the same location at the same time, we use the *NoOverlap* constraint on the activities that occur at the same location. This constraint ensures that transport activities of different trains for the same rail segment (location) do not overlap in time – the trains are not using that segment at the same time.

$$\forall j \in \{1, \dots, m\}: \quad NoOverlap(\bigcup_{T_1, \dots, T_n} Traverse(T_i, L_j, L_k) \tag{14}$$

3.3 Comparison of the Models

The two proposed models both abstract the train movement; however, there are slight differences between what is possible to model by each model. The

MAPF model can express the position of the whole train at each timestep very precisely. In contrast, the scheduling model only provides which location (rail segment) of the network is traversed at each time, but the train's exact location at that rail segment is not given. This degree of imprecision can be changed based on the granularity of the locations created.

The MAPF model allows the trains to visit any vertex in the grid any number of times. On the other hand, the scheduling model allows using each activity only once. This property means that the train cannot return to a given location with the same orientation. This issue can be solved by creating extra activities with a unique name for the given location to allow the train to return. A change to the following possible locations is then needed. However, it is usually not necessary in a real-world setting for a train to move in a loop to reach its destination.

While the last two properties were shortcomings of the scheduling model (the model does not support planning entirely), this model can represent broader scenarios than the classical MAPF model. For example, by setting the transfer times and duration of each activity, we can simulate different speeds for each train. Furthermore, we can impose different travel speeds for each location. Similarly, we can also simulate speeding up and slowing down the train in the locations leading to the stations. Also, required visited locations may be set by using the *PresenceOf* constraint. The exact time of visit is then left to the solver. On the other hand, if we know in advance the time window when this required visit needs to occur (for example, from the given timetable), we can also set it by changing the minimal start and maximal end times of the activity.

4 EXPERIMENTAL EVALUATION

To test the proposed modeling methods, we implemented the scheduling approach in the IBM CP Optimizer version 12.8 (Laborie, 2009). For the SAT-based approach we used the Picat language and compiler version 2.7b7 (Barták et al., 2017). The experiments were run on a PC with an AMD Ryzen 7 4700U running at 2.00 GHz with 16 GB of RAM. We used a cutoff time of 300 seconds per problem instance.

4.1 Instances

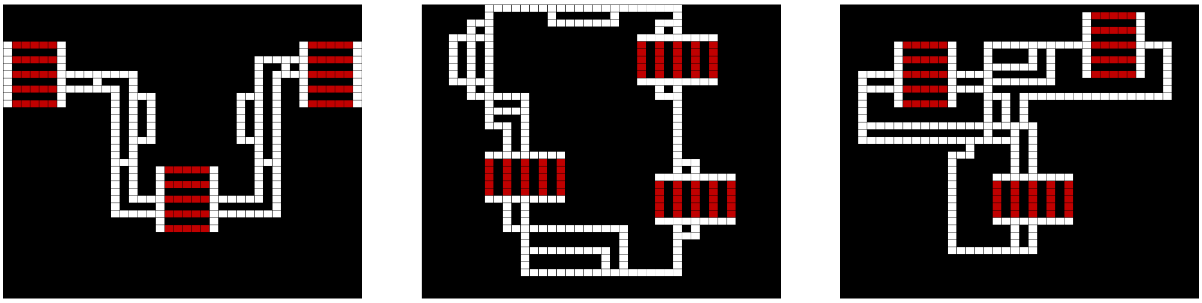
Classical MAPF techniques are frequently compared using grid-based benchmark maps (Stern et al., 2019). However, these maps are far from maps representing train networks. Flatland challenge maps (Mohanty et al., 2020) describe railway networks, but they are designed for unit-size agents and do not represent well stations with multiple platforms. Therefore, we created three 4-connected grid maps representing a structure of a rail network inspired by the Flatland challenge maps, but with more detailed train stations (see Figure 8). Each of the maps consists of three stations (represented by the red highlight), each with five rail tracks of length five units. The stations represent a location where a train may start and end after the execution of the found plan. In total there are fifteen possible starting and goal locations on each map. We created the maximal number of agents (fifteen) of random length between two and five for each map. The start location was placed at random and the goal location was also chosen by random from one of the other two stations. Note that the orientation of the train is also an important part of the start and goal location. This process was done five times with different random seeds. Each instance is then created by considering a different number of trains, from one to fifteen, on the given map. In total, this produces 225 different instances.

The representation for the MAPF model is quite straightforward. Each vertex in the map represents a location. Each train with length l consists of l agents with assigned start and goal locations such that the orientation of the train is maintained.

For the scheduling model, we first needed to split the grid map into locations. If each vertex was a single location, we would lose the strength of the model, so instead, we considered a sequence of vertices with only two neighbors as a single location. Example of such transformation of map *Linear* can be seen in Figure 9. Note that this transformation is only needed due to comparison with the MAPF-like approach. In reality, the railway network is given as a graph of tracks (Figure 5).

Over each of the locations, an activity was created for each train. The minimal start and maximal end of the activity were set to 0 and an upper bound on the whole plan. The duration of the activity was set as the length of the train plus the length of the location (i.e., the number of vertices in that location). The transfer time was set as the length of the train. Using clever preprocessing, the bounds on the start and end times of the activity may be improved, however, in our experiments, we did not find this to be the bottleneck for the solver.

Since the two models do not provide exactly the same solutions, as we discussed at the end of the last section, and we do not look for an optimal solution, we do not increase the upper bound on makespan T in the SAT model by 1. Rather, we increase the T



(a) Linear. (b) Circle. (c) Tree.

Figure 8: Maps used in the experiments.

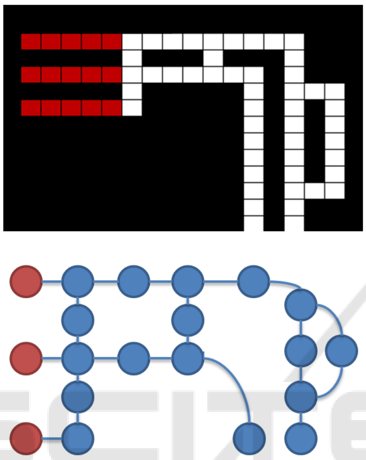


Figure 9: In the top is part of the *Linear* map. On the bottom is its representation as locations that the scheduling model uses. Note that the long sequences of vertices with just two neighbours are abstracted as a single location.

by the length of the longest train, which is five in our instances. The reasoning behind this is that if there is no solution in a given lower bound on makespan, it may be caused either by a conflict of two trains or by the fact that a train is required to arrive at the station from a different direction. Waiting for a train to free a given vertex takes the same number of timesteps as is the length of the train. Arriving at a station from a different direction means going around the station. In both cases increasing the T by just 1 would not help. Of course, this approach does not guarantee finding a makespan-optimal solution, but it should be able to find a solution in less computational time.

4.2 Results

Out of the 225 instances, the SAT MAPF model was able to solve only 38 with the most trains solved being 5 on map *Tree*. On the other hand, the scheduling model was able to solve all of the instances using just a fraction of the allowed runtime. Figure 10 shows the

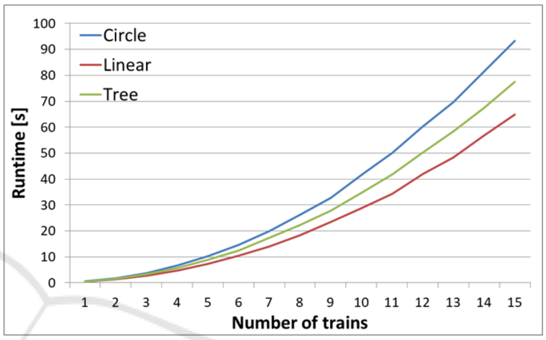


Figure 10: Measured runtime for the scheduling approach. The presented values are average runtime of instances with a given number of agents.

average runtime per number of trains. The results are split for the three different types of maps. From these results, we can see that the runtime increases with the number of trains being present in the graph, as is expected. We also observe that the different structure of the maps gives a clear ordering in hardness of the instances with *Linear* being the easiest, while *Circle* is the hardest. We can observe the same ordering in the SAT model results.

Such poor performance by the SAT-based model was not expected. We performed the experiments again without the constraint on agents modeling a train to remain connected. Thus a classical MAPF problem was being solved. This relaxation was able to solve 111 instances out of 225. This is a substantial improvement, however, it is still far away from solving all of the instances as the scheduling model is able to. From this result, we can draw the conclusion that the added constraints on the train movement are hard for the solver. Furthermore, we can see that this type of instance is hard even for classical MAPF. Indeed, when 13 trains are present (which was the maximal number solved in this experiment by the classical MAPF algorithm), there are up to 65 single agents on the map. The map itself provides a challenging environment where the agents do not have much space to avoid each other.

5 CONCLUSION

In this paper, we studied the problem of train routing in a shared environment. We proposed two models. The first one is based on a classical MAPF reduction-based approach. In this approach, each train is modeled as a group of agents moving to the goal location in a connected manner, simulating the position of the whole train. Our second model is inspired by a scheduling problem, where the graph is split into continuous locations – rail segments. Each activity represents a traversal of a given location by a train. The activities are intertwined in such a manner to ensure that the trains do not perform a forbidden movement; specifically, they do not occupy the exact physical location at the same time.

We compared the two models by their ability to model the real world. It seems that the scheduling model can represent a wider variety of real-world attributes, such as different speeds in any segment of the rail network. On the other hand, it is more complicated to create the instance for this model automatically from the grid-like MAPF maps

We also evaluated the models empirically and found that the MAPF model does not scale well over instances with just a few trains. On the other hand, the scheduling model solved all of the instances in just a fraction of the allocated runtime, showing great potential.

ACKNOWLEDGEMENTS

This research is supported by the Czech-USA Cooperative Scientific Research Project LTAUSA19072.

REFERENCES

Andreychuk, A., Yakovlev, K. S., Atzmon, D., and Stern, R. (2019). Multi-agent pathfinding with continuous time. In Kraus, S., editor, *28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 39–45. ijcai.org.

Atzmon, D., Diei, A., and Rave, D. (2019). Multi-train path finding. In Surynek, P. and Yeoh, W., editors, *12th International Symposium on Combinatorial Search (SOCS)*, pages 125–129. AAAI Press.

Atzmon, D., Felner, A., Stern, R., Wagner, G., Barták, R., and Zhou, N. (2017). k-robust multi-agent path finding. In Fukunaga, A. and Kishimoto, A., editors, *10th International Symposium on Combinatorial Search (SOCS)*, pages 157–158. AAAI Press.

Atzmon, D., Zax, Y., Kivity, E., Avitan, L., Morag, J., and Felner, A. (2020). Generalizing multi-agent path finding for heterogeneous agents. In Harabor, D. and Vallati, M., editors, *13th International Symposium on Combinatorial Search (SOCS)*, pages 101–105. AAAI Press.

Barták, R., Svancara, J., and Vlk, M. (2018). A scheduling-based approach to multi-agent path finding with weighted and capacitated arcs. In André, E., Koenig, S., Dastani, M., and Sukthankar, G., editors, *17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 748–756.

Barták, R., Zhou, N.-F., Stern, R., Boyarski, E., and Surynek, P. (2017). Modeling and solving the multi-agent pathfinding problem in picat. In *29th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 959–966. IEEE Computer Society.

Laborie, P. (2009). IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. In van Hoeve, W. J. and Hooker, J. N., editors, *6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, volume 5547 of *Lecture Notes in Computer Science*, pages 148–162. Springer.

Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2009). Reasoning with conditional time-intervals. part ii: An algebraical model for resources. In *22nd Florida AI Research Society Conference (FLAIRS)*, pages 201–206.

Li, J., Chen, Z., Zheng, Y., Chan, S., Harabor, D., Stuckey, P. J., Ma, H., and Koenig, S. (2021). Scalable rail planning and replanning: Winning the 2020 flatland challenge. In Biundo, S., Do, M., Goldman, R., Katz, M., Yang, Q., and Zhuo, H. H., editors, *31st International Conference on Automated Planning and Scheduling (ICAPS)*, pages 477–485. AAAI Press.

Mohanty, S. P., Nygren, E., Laurent, F., Schneider, M., Scheller, C., Bhattacharya, N., Watson, J. D., Egli, A., Eichenberger, C., Baumberger, C., Vienken, G., Sturm, I., Sartoretti, G., and Spigler, G. (2020). Flatland-rl : Multi-agent reinforcement learning on trains. *CoRR*, abs/2012.05893.

Stern, R., Sturtevant, N. R., Felner, A., Koenig, S., Ma, H., Walker, T. T., Li, J., Atzmon, D., Cohen, L., Kumar, T. K. S., Barták, R., and Boyarski, E. (2019). Multi-agent pathfinding: Definitions, variants, and benchmarks. In Surynek, P. and Yeoh, W., editors, *12th International Symposium on Combinatorial Search (SOCS)*, pages 151–159. AAAI Press.