Train Route Planning as a Multi-agent Path Finding Problem

Mauricio Salerno(✉), Yolanda E-Martín, Raquel Fuentetaja, Alba Gragera, Alberto Pozanco, and Daniel Borrajo

Universidad Carlos III de Madrid, Madrid, Spain
{msalerno,agragera,apozanco}@pa.uc3m.es, {yescuder,rfuentet}@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract. The train routing and timetabling problem consists of setting routes and schedules of a set of vehicles given their initial timetables and a railway network. The number of vehicles, the complexity and limited capacity of the railway network, and the time constraints make this problem difficult to solve. In this paper, we model this problem as a Multi-Agent Pathfinding problem, and propose a Conflict-Based Search approach to solve it. In our approach, we consider the complex properties found in this scenario such as continuous time, agents that function as convoys of arbitrary length, arbitrary action duration, and railway networks to find a solution. We analyze and discuss our approach explaining the main difficulties and evaluate it on several scenarios.

Keywords: Train routing and timetabling · Multi-agent path finding · Heuristic search

1 Introduction

The Train Routing Problem (TRP) consists of routing a set of vehicles in railway network and assigning them tracks to arrive (depart) to (from) the station. The Train Timetabling Problem (TTP) consists of scheduling a set of vehicles without violating track capacities and satisfying some time constraints. Both problems can be studied separately, but they are directly related: scheduling a train not only depends on the railway network and departure and arrival times, but also on the route and all possible conflicts with other trains' routes. The combination of both problems is difficult to solve because of the number of trains, the complexity and limited capacity of the railway network, and the time constraints.

Both problems have been widely studied using different techniques such as Integer Linear Programming (ILP) [3], Mixed Integer Programming (MIP) [11,13], multi-objective linear programming [12], local search [4,7], heuristic search [6], or Constraint Satisfaction Problems (CSP) [14]. In this work, we study the Train Routing and Timetabling problems together as Multi-Agent Pathfinding (MAPF) problems [5]. A MAPF problem is the problem of finding paths for a set of agents such that every agent reaches its goal while avoiding collisions.

E. Alba et al. (Eds.): CAEPIA 2021, LNAI 12882, pp. 237–246, 2021.
https://doi.org/10.1007/978-3-030-85713-4_23

TRP has been already studied by previous works using a MAPF approach [2]. However, they relax the problem by making assumptions that hinder the use of such techniques in more realistic scenarios as: discrete time steps, unit-cost actions, vehicles losing dimensionality while stopped, and grid scenarios. Andreychuk et al. [1] study the MAPF problem under a more realistic setting, where agents have volume, there are arbitrary cost actions, and continuous time. The type of agents considered are 2D non rotating agents situated also in grids.

Our work is inspired by Atzmon et al. [2] and Andreychuk et al. [1], but we go one step forward by considering: (1) vehicles that keep their size throughout the whole process; (2) use the concept of *resources* to determine conflicts, which allows dealing with collision detection; and (3) more realistic railway networks.

The rest of the paper is organized as follows. First, we introduce a formal definition of the Train Routing and Timetabling Problem (TRTP) we address. Then, we define the algorithms we use to solve the problem. Finally, we present an empirical study in different scenarios, and the conclusions and future work.

2 Problem Definition

We define the Train Routing and Timetabling Problem (TRTP), as a tuple $(\mathcal{I}, \mathcal{R}, \mathcal{V}, \mathcal{T})$ where $\mathcal{I}$ represents the railway topology, defined by a set of segments S and a set of points P; $\mathcal{R}$ is a set of resources; $\mathcal{V}$ is a set of vehicles; and $\mathcal{T}$ is the initial route timetable for all vehicles. Every segment $s \in S$ has a starting point and an end point (s_s, s_e), where $s_s, s_e \in P$ and it is associated to one of two directions $d_s \in \{1, 2\}$, a length l_s, and a travel time t_s. Bidirectional segments are considered as different segments with opposite directions. Resources are the basis of the railway safety system. They are exclusive since they can only be occupied by a single vehicle at a time. Thus, each resource is defined by a set of segments $R \subseteq S$, such that $\forall i, j \; i \neq j \rightarrow R_i \cap R_j = \emptyset$. Each vehicle $v \in \mathcal{V}$ has a length l_v. We assume all vehicles can travel in both directions and have constant speed c. A route timetable for a vehicle v is a sequence of tuples $\langle p, d, [t_a, t_d], o \rangle$, where p is the position of the vehicle's head, d is the direction, $[t_a, t_d]$ is the time interval defining the arrival and departure times of the vehicle head to/from p, and o is the sequence of segments (from head to tail) that the vehicle occupies. p can be defined by either a point, an area, or a platform area.

An *area* is a subset of segments $A \subseteq S$. A *platform area* is a subset of platform segments in an area, $T \subseteq A$. Platforms are the only segments where passengers can board and alight.

The initial route timetable corresponds to the official train timetable, which is a partial definition of the vehicles' route. It must specify boarding/alighting times in platform areas, which will be considered as hard constraints. Thus, for each vehicle it could specify: (a) either an entry or internal point of the infrastructure along with the arrival time to it; (b) one or more areas where the vehicle has to stop, or platform areas where the vehicle has to board/alight passengers, along with the corresponding times; and (c) an exit point from the infrastructure to the external railway network.

A solution to a TRTP is a route timetable that completely defines all vehicles' paths along with the arrival and departure point times, such that there are no conflicts among vehicles and all arrivals and departures are performed on-time.

The following definitions refer to concepts which are relevant for finding a solution to the TRTP.

Definition 1 (Stop time). *Given the arrival/departure interval* $[t_a^p, t_d^p]$ *for a specific point* p *in the route of a vehicle* v*, the stop time of* v *at* p *is defined as* $t_{stop}^p = t_d^p - t_a^p$*. If* $t_{stop} = 0$*,* p *is a non-stopping point.*

Vehicles have an arbitrary length, which could be larger than the length of the segment where their head is situated. This implies that a vehicle might occupy several segments at the same time. Thus, the solver needs to keep track of the segments' occupation as the route is being computed. Also, the solution must ensure that the route is safe, meaning that there are no collisions with other vehicles. In order to achieve that, the route must comply with resource exclusivity, which implies keeping track of the blocked resources and their releasing times.

Definition 2 (Occupied segments). *Let* $(p_0, \ldots, p_i)$ *be the points in the current route of a vehicle* v*, which determines a sequence of segments* $(s_0, \ldots, s_{i-1})$*, where each segment connects two consecutive route points. Let us assume that* v*'s head is situated at point* p_i*, which means that the vehicle is in segment* s_{i-1}*. Let* s_{i-k} *be the first segment from head to tail for which* $\sum_{j=1}^{k} l_{s_{i-j}} > l_v$*. Then,* v *occupies, from head to tail, the sequence of segments* $o = (s_{i-1}, \ldots, s_{i-k})$*.*

This definition is illustrated in Fig. 1 (left), where there is a vehicle whose head is at p_i, and the complete vehicle occupies the segments from s_{i-1} to s_{i-k}.

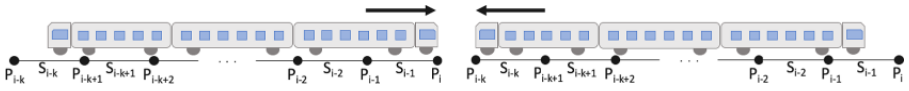

Fig. 1. Occupation and direction change scheme.

To ensure a solution that respects the railway safety system, we need to define the concept of *segment occupation time*, i.e., the time interval when a vehicle is occupying a segment. This interval starts when the vehicle's head arrives to the starting point of the segment and ends when the vehicle's tail leaves its end point. The upper bound of this interval is computed considering the head's departure time from the end point plus the time from head to tail. Formally:

Definition 3 (Segment occupation time). *Let* $(p_0, \ldots, p_i)$ *be the points in the current route of a vehicle* v*, and* $o = (s_{i-1}, \ldots, s_{i-k})$ *be the sequence of occupied segments. Then, the segment occupation time of* s_{i-k} *is defined by the time interval* $[t_a^{p_{i-k}}, t_d^{p_{i-k+1}} + t_{head\text{-}tail}^v + t_{delay}^o]$*, where* $t_{head\text{-}tail}^v = l_v/c$ *and* $t_{delay}^o = t_{stop}^{p_i} + t_{stop}^{p_{i-1}} + \cdots + t_{stop}^{p_{i-k+2}}$*.*

In this definition, the time from head to tail, $t^v_{head\text{-}tail}$, is computed considering the length of the vehicle and the constant speed c. The delay in leaving segment s_{i-k} is due to the (potential) stops of the vehicle's head in the points from p_{i-k+2} to p_i. As Fig. 1 (left) shows, these are the successive points where the vehicle can stop while its tail is still in segment s_{i-k}.

Resources are sets of exclusive segments. A resource is blocked when a vehicle enters in any of its segments [10]. When a resource is blocked, no other vehicles can use the resource until it is released. Note that a vehicle could block more than one resource at the same time. A resource is released when the vehicle's tail passes another resource plus a constant security time. Given the sequence o of occupied segments by a vehicle v, the list of blocked resources by v can be easily computed considering the segments in o and the resources they belong to. Given a resource $R = \{s_0, \dots s_n\}$, it would be blocked by a vehicle v within the interval $[t_{block}, t_{release} + t_{safe}]$, where t_{block} is the arrival time of v's head to a segment in R; $t_{release}$ is the time when v's tail leaves the resource; and t_{safe} is an input constant. The segment occupation times are computed following Definition 3.

In this work we assume vehicles can change their direction at a given moment. To represent a direction change in the route schedule, we replace the vehicle's tail with its head and vice versa. We consider the time to move the new vehicle's head to the nearest point in the opposite direction (referred as *setting time*) plus a *maneuver time*, t_m. Let $(p_i, d, [t^{p_i}_a, t^{p_i}_d], (s_{i-1}, \dots, s_{i-k}))$ be the last tuple in the route schedule for a vehicle before a direction change. Then, the next tuple in its schedule, after the direction change, will be $(p_{i-k}, \bar{d}, [t^{p_i}_d + t_m + t_{set}, t^{p_{i-k}}_d], o)$. p_{i-k} is the nearest point in the opposite direction, given that when the head was at p_i the vehicle was occupying the previous segments to s_{i-k}. The opposite direction is denoted as $\bar{d}$. The arrival time to this nearest point is the departure time from p_i, $t^{p_i}_d$, plus the setting and maneuver times. $t^{p_i}_d$ refers now to the tail departure since there was a direction change. The setting time is $t_{set} = (\sum_{j=1}^{k} l_{s_{i-j}} - l_v)/c$, where the numerator is the total length of the occupied segments minus the vehicle's length. This is the distance from the position of the vehicle's tail before the direction change to the point p_{i-k}, the nearest one in the opposite direction. The occupied segments o when the vehicle's head is at p_{i-k} are computed following Definition 2.

Figure 1 illustrates a direction change. The left figure shows the initial position of the vehicle before the direction change, where its head is located at point p_i (right direction). The right figure shows the final position of the vehicle after the direction change (left direction). The setting time is the time taken to move the vehicle's head from its actual position in segment s_{i-k} (left figure) to point p_{i-k}, which is the head position after the direction change (right figure).

3 Graph Representation of the Railway Network

In railway networks there are certain turns that vehicles cannot take due to physical properties of the track topology. Figure 2 (left) shows a small railway network where a train cannot turn from C to F if it reaches C from B (orange

train). However, the turn is possible if the train comes from D (grey train). Considering these physical restrictions when the network is represented as a graph would mean to deal with additional constraints to avoid impossible turns.

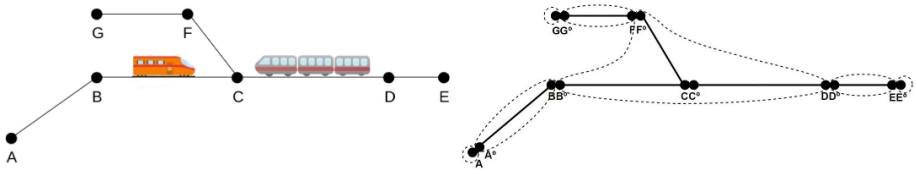

Fig. 2. Railway network with two vehicles (left) and corresponding DVG (right).

To overcome this issue, we represent the infrastructure as a Double Vertex Graph (DVG) [10], which consists of vertices (points) and edges (segments) that prevent impossible turns. We perform the search over this graph. The main idea of a DVG is that all vertices are duplicated, generating pairs (v, v°), where v and v° are joined vertices that represent the same point in the network. This association relies on a mapping function $\circ : V \mapsto V$ that for each vertex v returns its joined vertex $\circ(v)$ (or v°). This mapping function satisfies that $\circ(\circ(v)) = v$. During the search, reaching a vertex on the DVG implies to be moved automatically to its joined one, by applying the mapping function $\circ$. Therefore, the only allowed movements are those represented by the outgoing edges of the joined vertex. Outgoing edges do not include movements with impossible turns.

The infrastructure $\mathcal{I}$ is converted automatically into a DVG: each point v_i is converted into a double vertex (v_i, v_i°); each segment with direction d, defined by is starting and end points (v_s, v_e), generates the edge (v_s°, v_e); and each segment with direction $\bar{d}$, defined by points (v_s, v_e) generates the edge (v_s, v_e°). Figure 2 (right) shows the result of transforming the infrastructure in Fig. 2 (left) to a DVG. Thus, the infrastructure contains the segments: AB, BC, CD, DE, GF, FC (direction 1); BA, CB, DC, ED, FG, CF (direction 2).

A formal track topology, as defined by Montigel [10], includes both the DVG and the definition of resources, which guarantee safety in the railway system. Figure 2 (right) shows an example where resources represented with dashed lines. Considering the DVG, vertices connected by an edge belong to the same resource. Resource boundaries lie between joined vertices.

4 The TRTP as a MAPF Problem

In this section, we explain the approach followed to solve the TRTP problem as a MAPF problem. Specifically, we apply Conflict-Based Search (CBS) [15], where agents are trains and the search is performed over the formal track topology.

CBS consists of two search spaces: high-level and low-level. At the high level, the search is performed on a binary conflict tree (CT), which is created from

the found conflicts. At the low level, the search seeks a path for a single agent consistent with the imposed restrictions. The main idea behind this type of search is to allow each agent to find its own path at the low level while checking that those paths are conflict-free at the high level.

In this work, each node $n \in$ CT is a state that consists of: (1) a set of constraints (initially empty); (2) a potential solution to the TRTP, $\mathcal{T}_n$, consisting of complete route timetable for each vehicle; and (3) the total cost of the solution. To ensure the safety of the system, constraints are defined over resources; only one vehicle can be located at the same time in the same resource.

Definition 4 (Conflict). *A conflict appears when there is an overlap of the time intervals of two agents occupying the same resource. A conflict over a resource $R \in \mathcal{R}$ is defined as a tuple $\langle v_1, v_2, [t_1, t_2], [t_3, t_4] \rangle$, where $v_1, v_2 \in \mathcal{V}$ are the vehicles involved in the conflict, and $[t_1, t_2], [t_3, t_4]$ are the respective resource occupation time intervals with $[t_1, t_2] \cap [t_3, t_4] \neq \emptyset$.*

During the high-level search, we select the next node $n \in$ CT with least cost. If n does not have any conflict, n is a solution and we return the conflict-free route timetable for each agent on n. Otherwise, the search continues expanding n. A conflict in n is arbitrarily chosen and we constrain each agent's search according to the occupation time interval of the other agent involved in the conflict. We define two constraints for each selected conflict, represented as $c_1 = (v_1, R, [t_3, t_4])$ and $c_2 = (v_2, R, [t_1, t_2])$. The former represents that the agent v_1 must not be in R during the time interval $[t_3, t_4]$, preventing it from occupying R while v_2 is inside (c_2 is similar, but in the opposite direction). Then, we generate two new successors n_1 and n_2 in the CT, with the same set of constraints as n, plus the new one generated to each vehicle. Hence, n_1 will restrict the solution of v_1 with c_1 while n_2 will do it to v_2 with c_2. A low-level search is performed for each node to find a new solution consistent with the new constraints.

The low-level search seeks a complete route timetable for a vehicle consistent with its initial timetable $\mathcal{T}_0$, and its constraints. We conduct a modified A*, using as heuristic the minimum travel time without considering changes of direction. A state is represented as $\langle p, d, t_a, T \rangle$, where p is the vehicle's position, d is the direction the vehicle is facing, t_a is the arrival time to p, and T is the vehicle's timetable (partial route used to determine the vehicle's current occupation). Applicable actions are moving actions from one point to another. Stops happen at the arrival vertex and the mapping function $\circ$ is applied on departure. Direction changes can be applied at any moment (but have lower priority).

Sub-goals of a vehicle can be either points or areas. If a sub-goal g is a point, the search can be easily guided using a heuristic function that estimates the cost of reaching that point from the current point. However, when g is an area this is not enough because each area has a set of n predefined platform segments. Any path that leads the vehicle to one of these segments within the specified time interval could be a valid solution. Then, for that case, we first perform a platform selection step by generating as many search nodes as platforms are in the area. All of these nodes have the same state, but differ in the goal point.

The search is performed as follows. For each node n, $g(n)$ is the t_a to the current point, defined in its state. We keep two open lists during the search. The first one stores nodes n that do not require a direction change to reach the goal. These are expanded according to their $f(n) = g(n) + h(n)$ value. The second one stores nodes that require at least a direction change. They are sorted according to $g(n)$, but only expanded when the first list is empty. This guarantees that direction changes are only performed when necessary. Applying a move action from point p to point p' implies: computing the departure time t_d from the current point p; generating a successor whose partial route includes p along with its time interval; and computing the arrival time, t'_a of the next current point p'. We do not consider stops unless this option violates a constraint. Thus, we set t_d equal to t_a whenever possible. The arrival time at p' is computed as the departure time from p plus the travel time $t_{pp'}$ of the current traveling segment, $t'_a = t_d + t_{pp'}$. If the node being expanded is a goal node, the departure time is set to ∞: this indicates that the vehicle will remain parked in the current segment(s). Then, the solution is returned. A stop is a t_d such that $t_a < t_d$. We only consider them when they are needed to avoid violating any constraint.

Definition 5 (Constraint Violation). *Given a state $s = \langle p, d, t_a, T\rangle$ and a successor state $s' = \langle p', d', t'_a, T'\rangle$, generated by applying a move action from p to p' to the current vehicle v, a constraint $c = (v, R, [t_i, t_j])$ for v is violated if the segment (p, p') belongs to R and either: (a) $t_i < t_d < t_j$, or (b) $t_d < t_i$ and $t'_a > t_i$, or (c) $t'_a < t_i$, the successor s'' of s' involves a movement through a segment which still belongs to R, and this movement generates a violation of c.*

In (a), the violation can only be solved in two ways: (1) making $t_d < t_i$, or (2) making $t_d > t_j$. Assuming that the vehicle does not stop at p (i.e. $t_d = t_a$ and $t'_a = t_a + t_{pp'}$), there is no chance to make $t_d < t_i$ since t_d is the earliest departure time. Therefore, we choose to solve the constraint violation making $t_d > t_j$ by setting $t_d = t_j + t_{safe}$, where t_{safe} is the constant security time defined for the network. Then, $t'_a = t_d + t_{pp'}$. It implies making a stop at p to delay entering the resource until the upper bound of the forbidden interval is reached. In (b), the solution is similar since an earlier departure from p would be needed to achieve $t'_a < t_i$. This is not possible because t_d is the earliest departure time (if the vehicle does not stop at p). In (c), the constraint violation cannot be checked at the expansion time of the current node (state s). At this point, it is a potential violation since we do not know neither the departure time in s' nor the arrival time in s'' until s' is expanded. In such scenario, we must create two child nodes to guarantee completeness. One is the current successor ($t_d = t_a$), which does not violate the constraint at the moment. The other node is created by making a stop at p, to delay its departure time to the upper bound of the forbidden interval. This is done by following (a) or (b). When the departure time in s' or the arrival time in s'' violates the constraint, the path is discarded.

This conflict resolution approach is also valid in those cases where a direction change is applicable (a direction change action generates an additional successor). In all cases, when selecting the departure time, the occupied segments, and their occupation times are computed following Definitions 2 and 3.

The final route timetable generated is optimal with respect to the travel time while complying with the imposed constraints. Considering the way constraints are created, the solution might not be optimal. Constraints are added using the vehicle's occupation time in the resource, not the minimum occupation time possible for that resource. So a solution that complies with the constraints might be more costly than one with different constraints.

5 Evaluation

We tested 3 networks, following experts' guidelines, of increasing size: 110, 148, and 312 segments, with 40, 55, and 115 resources respectively. We ran experiments with an increasing number of agents k, ranging from 2 to 10. The maximum number of agents seems to be low, but the networks are relatively small. Hence, we are testing the very complex scenarios due to a high percentage of the network occupied by agents. This parameter, denoted as %O, is computed as the number of initially occupied resources divided by the total number of resources in the network. We randomly assigned a different length to each agent from 5 to 25. For all networks, the average segment length is 20, so agents might occupy two segments in some cases. Each agent has associated a platform as goal.

We tested with three types of deadlines. To compute them, for each network, we ran 1000 problems with one agent having a random initial position and destination. The base deadline, b_d, is set to the maximum time needed for the agent to reach its destination among all runs. b_d is considered as a hard deadline in practice, since it is very unlikely the algorithm finds a solution for all agents that takes less time than b_d. The higher the number of agents in the network, the higher the number of conflicts, which causes delays over all vehicles. The other two deadlines are: $d = b_d \times 2$ (medium deadline); and $d = b_d \times 4$ (soft deadline).

We generated 100 random problems and ran them for each type of deadline and number of agents. We gave 60 s to the algorithm to solve each instance and report: (1) if the problem was solved within the time bound, S; (2) the sum of costs of all the agents' plans, SOC; (3) the makespan, MK, which is the time step at which the last agent reaches its destination; and (4) the time T in milliseconds needed to solve the problem.

Table 1 shows the results[1]. For the small network, we only report results up to 6 agents since with a higher number of agents none of the problems were solved within the time bound. A higher number of agents makes the small network intractable with very high occupancy rates, which prevent agents from reaching their destinations. For all the networks, increasing the number of agents causes higher occupancy rates and lower success rates, as expected. Having more agents in the network means more potential conflicts to be solved, many of them being unsolvable within the given time bound. The success rate drops below 0.5 when the occupancy of the network is 25% or higher in the small and medium networks. For the large network, there are worse success rates for lower occupancy rates. The topology of the network creates a bottleneck in some resources that connect

[1] Results obtained on an Intel Core i7 2.9 GHz CPU computer with 16 GB of RAM.

Table 1. Results for the three types of networks.

			Soft deadline				Medium deadline				Hard deadline			
Network	k	%O	S	SOC	MK	T	S	SOC	MK	T	S	SOC	MK	T
SMALL	2	10.0	1.0	122.1	75.8	184.5	0.9	120.2	77.1	147.6	1.0	120.9	76.9	94.7
	3	15.0	0.9	174.1	84.2	500.2	0.8	167.5	86.6	581.5	0.9	170.9	86.3	1494.5
	4	20.0	0.8	235.1	98.1	1722.4	0.7	226.7	90.5	3186.1	0.8	222.3	89.1	3143.4
	5	25.0	0.5	256.8	93.3	2437.0	0.5	261.5	92.5	2430.5	0.6	279.8	97.3	6310.1
	6	30.0	0.3	289.5	89.6	2464.9	0.2	302.2	101.1	6784.7	0.3	292.0	98.7	10046.5
MEDIUM	2	7.2	1.0	115.6	74.2	122.2	0.9	116.2	74.8	105.7	0.9	121.7	77.6	81.5
	3	10.9	0.9	163.4	79.8	830.8	0.9	165.9	80.7	1056.1	0.9	169.5	80.5	1644.6
	4	14.5	0.9	209.9	83.1	1281.1	0.8	209.8	85.3	1365.9	0.9	209.1	81.9	1640.2
	5	18.1	0.7	256.8	84.6	3092.4	0.8	240.9	84.6	4122.2	0.8	242.9	81.3	2955.7
	6	21.8	0.5	297.6	91.6	7713.5	0.6	283.3	92.3	4579.2	0.7	271.6	81.9	5395.1
	7	25.4	0.3	293.8	79.6	8366.5	0.4	308.3	81.0	7828.9	0.5	300.6	86.7	10867.0
	8	29.0	0.2	359.6	97.8	10634.9	0.1	355.6	84.6	6660.3	0.2	330.7	88.7	8672.1
	9	32.7	0.07	326.5	106.5	11896.5	0.1	356.1	87.7	13187.9	0.09	324.0	79.5	23325.1
	10	36.3	0.05	390.5	89.0	28865.7	0.08	365.3	93.1	14941.0	0.1	378.5	90.3	23050.8
LARGE	2	3.4	0.9	148.1	95.5	90.1	1.0	153.5	96.4	235.6	0.9	163.8	102.7	189.4
	3	5.2	0.9	226.5	111.2	244.7	0.9	226.9	109.0	477.4	0.9	221.0	106.8	680.37
	4	6.9	0.9	282.7	111.8	990.2	0.9	300.6	117.1	1697.6	0.9	292.4	115.2	2321.1
	5	8.7	0.8	344.7	119.4	2909.1	0.8	349.3	120.8	4069.4	0.8	337.1	118.0	1751.0
	6	10.4	0.6	401.3	125.8	5492.0	0.7	404.9	122.1	7108.1	0.7	408.3	124.0	4687.2
	7	12.1	0.5	446.7	118.2	8220.7	0.5	484.4	126.8	9653.6	0.4	452.6	122.1	6645.8
	8	13.9	0.2	497.5	123.5	10284.9	0.2	511.7	129.2	14845.7	0.3	503.5	135.5	10484.7
	9	15.6	0.1	509.1	117.9	26109.0	0.2	487.4	117.8	17595.5	0.2	508.9	116.1	14108.9
	10	17.3	0.1	555.2	121.5	13578.3	0.08	585.1	118.4	9409.1	0.07	601.6	131.3	14904.5

two main areas, which causes a high number of conflicts. Regarding the deadlines, there is no apparent relationship between the type of deadline and the success rate. Looser deadlines have a less constrained search space, and the algorithm might exceed the given time while looking for a solution.

6 Conclusions and Future Work

In this paper we introduced an approach for solving TRTP problems as MAPF problems using CBS. We consider specific problem's features: the search is performed in a DVG, which allows representing implicitly the physical characteristics of railway networks and exclusive resources related to the railway safety system; continuous time, deadlines, and actions with duration; agents with arbitrary length; direction change maneuvers; and abstract goal definition that may involve solving the platforming problem.

We evaluated the approach on three different scenarios, varying the size of the network, the number of agents, and the deadline tightness. The results show that the higher the occupancy rate is, the lower the success rate is, given that a higher number of agents results in a higher number of potential conflicts.

We would like to consider the full complexity inherent to a real TRTP, which would involve dealing with sub-goal sequences in the initial route timetable, vehicle coupling and decoupling maneuvers, resource sharing, deposit operations, and

additional metrics to consider the platforming occupation time or the robustness of timetables. We would also like to evaluate it in real-world scenarios.

Acknowledgements. This work was funded by research projects TIN2017-88476-C2-2-R, RTC-2017-6753-4 of Spanish Ministerio de Economía, Industria y Competitividad/FEDER UE, and the Madrid government under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17), V PRICIT (Regional Programme of Research and Technological Innovation). This work was developed in cooperation with Goal Systems S.L. (www.goalsystems.com) whose working team provided expert knowledge of railway management and its properties. Special thanks to the technical staff for making this cooperation possible.

References

1. Andreychuk, A., Yakovlev, K., Atzmon, D., Stern, R.: Multi-agent pathfinding with continuous time. In: Proceedings of IJCAI-19, pp. 39–45 (2019)
2. Atzmon, D., Diei, A., Rave, D.: Multi-train path finding. In: Proceedings of SOCS-19, pp. 125–129 (2019)
3. Cacchiani, V., Furini, F., Kidd, M.: Approaches to a real-world train timetabling problem in a railway node. Omega **58**, 97–110 (2015)
4. Dewilde, T., Sels, P., Cattrysse, D., Vansteenwegen, P.: Robust railway station planning: an interaction between routing, timetabling and platforming. J. Rail Transp. Plan. Manag. **3**, 68–77 (2013)
5. Felner, A., et al.: Search-based optimal solvers for the multi-agent pathfinding problem: summary and challenges. In: Proceedings of SOCS-17, pp. 29–37 (2017)
6. Flórez, J., Torralba, A., Borrajo, D., Linares López, C., Olaya, A., Sáenz, J.: Combining linear programming and automated planning to solve intermodal transportation problems. Eur. J. Oper. Res. **227**, 216–226 (2013)
7. Higgins, A., Kozan, E., Ferreira, L.: Heuristic techniques for single line train scheduling. J. Heuristics **3**, 43–62 (1997)
8. Li, J., Surynek, P., Felner, A., Ma, H., Kumar, T.S., Koeing, S.: Multi-agent path finding for large agents. In: Proceedings of AAAI-19, pp. 7627–7634 (2019)
9. Ma, H., Wagner, G., Felner, A., Li, J., Kumar, T., Koenig, S.: Multi-agent path finding with deadlines. In: Proceedings of IJCAI-18, pp. 417–423 (2018)
10. Montigel, M.: Representation of track topologies with double vertex graphs. In: Computers in Railway, vol. 2 (1992)
11. Murali, P., Ordóñez, F., Dessouky, M.: Modeling strategies for effectively routing freight trains through complex networks. Transp. Res. Part C Emerg. Technol. **70**, 197–213 (2016)
12. Pouryousef, H., Lautala, P., Watkins, D.: Development of hybrid optimization of train schedules model for n-track rail corridors. Transp. Res. Part C Emerg. Technol. **67**, 169–192 (2016)
13. Qi, J., Yang, L., Gao, Y., Li, S., Gao, Z.: Integrated multi-track station layout design and train scheduling models on railway corridors. Transp. Res. Part C Emerg. Technol. **69**, 91–119 (2016)
14. Rodríguez, J.: A constraint programming model for real-time train scheduling at junctions. Transp. Res. Part B Methodol. **41**, 231–245 (2007)
15. Sharon, G., Stern, R., Felner, A., Sturtevant, N.: Conflict-based search for optimal multi-agent pathfinding. Artif. Intell. **219**, 40–66 (2015)