Optimal In-Station Train Dispatching via Symbolic Pattern Planning

Matteo Cardellini[1*], **Enrico Giunchiglia**[1], **Davide Anguita**[1], **Carmelo Lofiego**[2], **Luca Oneto**[1], **Pietro Ratto**[2],

[1]Università degli Studi di Genova, Genova, Italy
[2]Hitachi Rail STS, Genova, Italy
{matteo.cardellini, enrico.giunchiglia, davide.anguita, luca.oneto}@unige.it,
{carmelo.lofiego, pietro.ratto}@hitachirail.com

Abstract

The optimal In-Station Train Dispatching (InSTraDi) problem consists in commanding the movements of trains inside a railway station while both (i) respecting safety, time, and travel constraints and (ii) minimizing delays. In Symbolic Pattern Planning (SPP), a pattern, suggesting the sequence of actions to reach the goal, is encoded in a logic formula whose models correspond to valid plans. If no valid plan is found, the pattern is extended until it covers a valid plan. However, plans of better quality could exist if we had continued extending the pattern. In this paper, we formalize the InSTraDi problem as a Temporal Planning Task with Intermediate Conditions and Effects, and we show an InSTraDi-dependent way to construct, in polynomial time, a pattern ensuring the optimal plan can be found by the SPP approach without never extending the pattern. Analysis on realistic railway data validate our approach.

1 Introduction

Railways, particularly in Europe, are essential for commuting millions of passengers daily. Unfortunately, the increase in demand, especially after the pandemic and with oil prices rising, is rapidly congesting the network (Marcysiak and Marcysiak 2023). Stations are the only places where optimisation and resource management is possible, since, between stations, trains' overtake is rarely possible (Lindfeldt 2010). The *In-Station Train Dispatching (InSTraDi) Problem* (Zwaneveld et al. 1996; Cardellini et al. 2021) consists in *dispatching* the movements of trains in a station while respecting constraints of safety (e.g., avoid trains' collision), time (e.g., trains not departing before their scheduled time) and travel (e.g., trains must exit towards their destination). An optimal *dispatchment* is the one that minimize delays w.r.t. a predetermined timetable. The problem is NP-hard (Zwaneveld et al. 1996), and for this reason, has mostly been approached by Mixed-Integer Linear Programming (MILP) (Billionnet 2003; Mannino and Mascis 2009; Lamorgese and Mannino 2015; Lamorgese et al. 2018) or Constraint Programming (CP) (Rodriguez 2007; Schutt et al. 2025) techniques.

Planning is one of the oldest problems in AI (McCarthy and Hayes 1969) and tells the tale of *agents* (in our case trains) whose *goal* is to reach a desired state from an *initial* one through *actions*. To solve a planning task, the *planning as satisfiability* (PaS) approach (Kautz and Selman 1992), transforms the problem into a logic formula (Barrett and Tinelli 2018) whose models corresponds to plans of length n, starting from $n = 0$, increasing n upon failure. Symbolic Pattern Planning (SPP) (Cardellini, Giunchiglia, and Maratea 2026) is a novel approach based on PaS where a *pattern* $\prec$, i.e., a finite sequence of actions, is encoded in a SMT formula to suggest a causal order between the actions. A *simple* and *complete* pattern $\prec_I$, i.e., containing each action exactly one, is computed in the initial state, and, if the suggestion provided by $\prec{=}\prec_I$ is inaccurate and no valid plan can be found, $\prec$ is *extended* by concatenating $\prec_I$ to it, and the procedure is iterated until $\prec$ contains a valid plan. A limitation of SPP is that, once a plan is found with $\prec$, we cannot guarantee its optimality, since better solutions could exist at further extensions of $\prec$ with $\prec_I$. Deciding whether a plan exists for a given planning task is, in general, not easier than PSPACE-hard (Bylander 1991; Gigante et al. 2022a). Even if planning can solve problems harder than InSTraDi– and thus seems an overkill – it has already been successfully applied to it (Cardellini et al. 2021). Indeed, the main advantage of the planning formalism is its *action-centricity* (Gelfond and Lifschitz 1998), which makes the planning task readable even for stakeholders without planning expertise – not true for MILP and CP encodings.

In this paper, we formalize the optimal InSTraDi problem and encode it as a Temporal Planning Task with Intermediate Conditions and Effects (ICEs) (Valentini, Micheli, and Cimatti 2020), a fragment of planning for which SPP has been recently extended (Cardellini and Giunchiglia 2025; Cardellini and Giunchiglia 2026). We present a InSTraDi-dependent way to compute a pattern guaranteeing 1-*completeness*, i.e., all plans (among them, the optimal) can be found by SPP without never needing to extend the pattern. Indeed, we exploit the following properties of InSTraDi: (i) train's movements cannot be repeated and have a predetermined order (i.e., arrive, enter, stop, depart and exit), (ii) there is a precedence between trains entering/exiting the station from/to the same line. We show three ways to compute patterns while guaranteeing 1-completeness, each producing patterns of different size. In the third approach, we introduce domain knowledge, pruning away some very unlikely plans (e.g., where a train waits for another train arriving much later), but greatly improving performances and empirically maintaining optimality. We analyse our approach

*Corresponding Author

on benchmark scenarios coming from a real station in Italy, validated by the Italian Railway Network's company and the major worldwide company for train signalling.

2 Preliminaries

We now outline the formalisms behind our approach: the INSTRADI problem, Temporal Planning with ICEs, and SPP.

2.1 In-Station Train Dispatching

A *station*'s components are the following.

Signals gives instructions to train conductors on whether to stop or proceed. In Fig. 1, train `r` is facing signal `22`.

Track circuits or *bits* are the smallest unit forming rails. Usually, bits are tens of meters long and are the smallest component detecting trains' position. In Fig. 1, train `r` is occupying bits `114`, `115`, `116`.

Platforms are sets of bits where trains can stop to board/alight passengers. A platform is delimited by two signals: *left* and *right*. In Fig. 1, platform `II` is composed of bits `114`, `115`, `116` and is delimited by `22` and `42`.

Entry and Exit points are signals bounding the station, e.g., in Fig.1, entry points `01` and `03` or exit points `02` and `04`.

Routes are sequences of bits connecting two signals, e.g. in Fig. 1, route `03-23` connects `03` with `23` through the sequence `109`; `108`; `119`; `118`; `124`; `123`.

Formally, a station is a tuple $Z = \langle S, B, P, E^+, E^-, R\rangle$, where S is a set of signals, B is a set of bits and P is a set of platforms. The function $\mathrm{bits}\colon P \mapsto 2^B$ maps a platform to its bits. The set $S_p \subseteq S$ denotes the signals delimiting a platform $p \in P$. The sets $E^+ \subseteq S$ and $E^- \subseteq S$ are the entry and exit points, respectively. A route $r \in R$ is a sequence $b_1; \ldots; b_m$, with $m > 0$, of bits. Let $s_1 = \mathrm{from}(r) \in S$ and $s_2 = \mathrm{to}(r) \in S$ be signals associated with route r, we say that r *connects* s_1 and s_2, r *originates* from s_1, and r *terminates* in s_2. No route originates from exit points nor terminates in an entry point. A *station graph* has signals as nodes and routes as edges. Nodes corresponding to signals in $E^+ \cap E^-$ are split in two: one where routes originate and one where they terminate. The station graph is acyclic by construction. Let $r, r' \in R$ be two distinct routes: if r and r' have one bit in common, we say that r and r' *intersect*.

Trains have an ID, identifying their trip through several stations, and an *objective* for each station. For example, in Italy, train `8620` moves every day from *Roma Termini* to *Milano Centrale* transiting from *Camogli* and stopping at *Genova Brignole*. A train can have one of four objectives: (i) *origin* (O), e.g., `8620` in *Roma Termini*, i.e., the train is at a platform and has to leave the station through an exit point, (ii) *destination* (D), e.g., `8620` in *Milano Centrale*, i.e., the train enters the station from an entry point and has to stop at a platform, (iii) *stop* (S), e.g., `8620` in *Genova Brignole*, i.e., the train passes through the station from entry to exit point and is required to stop at a platform, and (iv) *transit* (T), e.g., `8620` in *Camogli*, i.e., the train has only to pass through the modelled station. Formally, T is a set of trains. We define the functions $\mathrm{id} : T \mapsto \mathbb{N}$ mapping a train to its ID, $\mathrm{orig} : T \mapsto S$ and $\mathrm{dest} : T \mapsto S$ mapping a train to its origin and destination, $\mathrm{stops} : T \mapsto 2^P$ mapping a train to its (possibly empty) allowed stopping platforms, and $\mathrm{obj} : T \mapsto \{\mathrm{S}, \mathrm{T}, \mathrm{O}, \mathrm{D}\}$ mapping a train to its objective. We denote by $R_\tau \subseteq R$ all the routes (i.e., edges) of the station graph constituting a path between $\mathrm{orig}(\tau)$ and $\mathrm{dest}(\tau)$.

A *timetable* specifies when a train is expected to arrive and depart from a platform. Formally, a timetable is a pair of functions $\eta = \langle \mathrm{arrive}, \mathrm{leave}\rangle$ where arrive and $\mathrm{leave}\colon T \mapsto \mathbb{Z} \cup \{-\infty, +\infty\}$ map each train to the relative time in which it is scheduled to arrive to/depart from a platform, respectively. An origin train τ_{O}, a destination train τ_{D}, a transit train τ_{T} are such that $\mathrm{arrive}(\tau_{\mathrm{O}}) = -\infty$, $\mathrm{leave}(\tau_{\mathrm{D}}) = +\infty$ and $\mathrm{arrive}(\tau_{\mathrm{T}}) = \mathrm{leave}(\tau_{\mathrm{T}})$.

Simulation of the timings of the movements of trains inside the station would require complex characterizations of bits, platforms, and trains' length, together with trains' weight, position, and speed. Instead, in this work, we suppose to be provided with a *forecast system* able to estimate, for each train, the running times of bits and routes, the stopping time at platforms and the estimated arrival of trains at entry points. The forecast system could be implemented in different ways, from simple statistical methods or very cutting-edge Machine Learning technologies (Boleto et al. 2021). Formally, a *forecast* is composed of functions $F = \langle f_M, f_S, f_D, f_E\rangle$ where $f_M\colon T \times R \times B \mapsto \mathbb{N}$ estimates the time for a train to run through a bit in a route, the functions f_S and $f_D\colon T \times P \mapsto \mathbb{N}$ estimate the time to stop and the time to depart a platform (for a train τ with $\mathrm{obj}(\tau) = \mathrm{T}$, $f_S(\tau, p) = 0$ for each $p \in P$), and $f_E\colon T \mapsto \mathbb{Z}$ estimates the arrival time of a train at an entry point, if the estimation is negative, the train has already arrived. We denote the total running time of a route $r = c_1; \ldots; c_m$ as $f_M(\tau, r) = \sum_{i=1}^{m} f_M(\tau, r, c_i)$.

We now describe trains' movements in the station. Consider train `r` in Fig. 1. The train is *locking* the bits `114`, `115` and `116` composing platform `II`. Before the train moves away from platform `II`, the train must *reserve* a route originating from `22`. This reservation can be done if all the bits of the route are *unlocked*, i.e., no other train has reserved them, and causes their lock. After reserving a route, the train can *move* through it. While moving, the train unlocks the bits it passes, making them available again for other trains. For example, `r` can leave the station with route `22-02` after ensuring that all the bits are unlocked.

A tuple $\sigma = \langle \mathrm{lock}, \mathrm{reserve}, \mathrm{on}, \mathrm{occupy}, \mathrm{stopped}\rangle$ is a *station state* where $\mathrm{lock}\colon T \mapsto 2^B$ maps a train to the bit it is locking, $\mathrm{reserve}\colon T \mapsto 2^R$ maps a train to its reserved routes, $\mathrm{on}\colon T \mapsto 2^R$ maps a train to the routes it is moving on, occupy and stopped are partial functions $T \mapsto P$ mapping a train to the platform it is (possibly) occupying and has (possibly) stopped, respectively. A station state is *valid* if (i) a train moving through a route is also reserving it, (ii) routes reserved by a train are contiguous, (iii) if reserving a route, its locked bits are contiguous and the last bit of the route is locked, (iv) if a bit is locked, it is part of a reserved route or an occupied platform, (v) if a train is stopping, it is in its allowed stops, (vi) if a platform is occupied, all the bits in it are locked, and (vii) at most one train is locking a bit.

A *command* is an instruction describing how a train

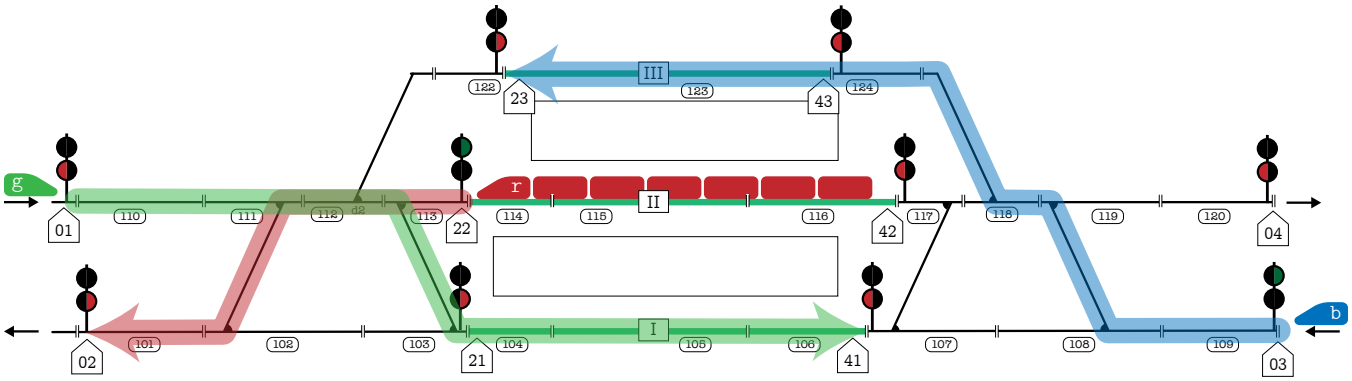


Figure 1: The benchmark station with three trains, r, g and b. The train r is stopping at platform II oriented towards the green signal 22 which allows entering route 22−02 to exit the station from the exit point 02. Train b (resp. g) is about to enter the station from the entry point 03 (resp. 01) using route 03−23 (resp. 01−41) to reach platform III (resp. I).

changes a station state. The commands can be: $\texttt{lock}(\tau, b)$, $\texttt{unlock}(\tau, b)$, $\texttt{reserve}(\tau, r)$, $\texttt{release}(\tau, r)$, $\texttt{move}(\tau, r)$, $\texttt{leave}(\tau, r)$, $\texttt{occupy}(\tau, p)$, $\texttt{free}(\tau, p)$ concerning a train $\tau \in T$, operating on a bit b, a route r or a platform p.

Let σ be a station state, α a set of commands, and σ' the station state obtained *applying* α *to* σ. *The transition from* σ *to* σ' *through* α *is valid* iff (i) σ and σ' are valid, (ii) routes are reserved or released in σ' contiguously with σ, (iii) if a route is reserved in σ, bits can only be unlocked in σ', (iv) the moved routes are contiguous with σ, (v) a train can stop in σ' in p only if it was locking $\text{bits}(p)$ in σ.

The station's entry/exit points are connected to other stations' exit/entry points via tracks called *line points*. Since line points are uninterrupted, overtakes can only happen in the station, and thus, an order between trains when exiting the station avoids, e.g., high-speed trains lagging slower ones. Moreover, from an entry point trains enter in a predetermined order. A *train order* is a tuple $o = \langle o^+, o^- \rangle$ where o^+ is the *entry order* and o^- is the *exit order*, both functions $T \times T \mapsto \{-1, 0, 1\}$. Let τ_1, τ_2 be two trains. If $o^+(\tau_1, \tau_2) < 0$ (resp. > 0), then τ_1 must enter the station before (resp. after) τ_2, if $o^+(\tau_1, \tau_2) = 0$ the order is irrelevant. The same holds for o^-. We assume that the functions in o induce a partial order over the trains in T. We denote by $T_o^+(\tau)$ the set of trains $\{\tau' \mid \tau' \in T, o^+(\tau', \tau) < 0)\}$, i.e., trains that enter before τ. We similarly define $T_o^-(\tau)$.

Finally, a *dispatchment* is a sequence of *timed sets of commands* $D = \langle t_1, \alpha_1 \rangle; \ldots; \langle t_k, \alpha_k \rangle$. Starting from a state σ_0 in $t_0 = 0$ the dispatchment induces a sequence of station states $\langle t_0, \sigma_0 \rangle; \langle t_1, \sigma_1 \rangle; \ldots; \langle t_k, \sigma_k \rangle$ where σ_i is the result of applying the commands' set α_i in σ_{i-1}, for $i \in [1, k]$. The dispatchment is *valid* if each transition from σ_{i-1} to σ_i through α_i is valid, for $i \in [1, k]$. D *respects a forecast* F if the times of the commands in D respect the estimates in F.

We are now in the position to define the *In-Station Train Dispatching* (INSTRADI) task $X = \langle Z, T, \eta, \sigma_0, F, o \rangle$. We are provided with a station Z, a set of trains T, a timetable η, the current state of the station σ_0, a forecast F, and a train order o. We assume that in σ_0 the station is *idle*, i.e., trains are only stopping at platforms, or they have not yet entered the station. We are tasked to find a valid dispatchment D from σ_0 which respects F guaranteeing that (i) each train never leaves the platform before the time specified in the timetable, (ii) each pair of trains respects the provided train order, and (iii) each train τ respects its objectives in the final state, i.e., if $\text{obj}(\tau) \in \{\text{S}, \text{T}, \text{O}\}$ then τ has left the station from $\text{dest}(\tau)$, if $\text{obj}(\tau) = \text{D}$, then τ is stopping, and if $\text{obj}(\tau) = \text{S}$, then τ has stopped. Let $\text{realLeave}(D, \tau)$ be the time associated to the command $\texttt{free}(\tau, p)$ in D for any platform p. We denote the *cost of* D as[1]

$$cost(D) = \textstyle\sum_{\tau \in T} \text{realLeave}(D, \tau) - \text{leave}(\tau), \qquad (1)$$

where $\text{leave}(\tau)$ is the departure time in the timetable η. We say that D is *optimal* if there is no another dispatchment D' for X s.t. $cost(D') < cost(D)$. Each summand in Eq. 1 is positive, since a train τ cannot depart before $\text{leave}(\tau)$.

2.2 Temporal Planning with ICEs

A *temporal planning task with Intermediate Conditions and Effects* (ICEs) (in the following just "*planning task*") is a tuple $\Pi = \langle V, A, I, G, C, E \rangle$, where V is a set of *propositional* variables, with domain in $\{\top, \bot\}$. We denote by $\text{lit}(V) = \{v, \neg v \mid v \in V\}$ the set of *literals of* V. We say that a set $L \subseteq \text{lit}(V)$ is *consistent* if, for each $v \in V$, $\{v, \neg v\} \not\subseteq L$. In the planning task Π, the set A is a finite set of *durative actions*. A durative action b is a tuple $\langle \text{icond}(b), \text{ieff}(b), [L, U] \rangle$, where $\text{icond}(b)$ and $\text{ieff}(b)$ are sets of *action intermediate conditions* and *action intermediate effects* of b, while $L, U \in \mathbb{Q}^{\geq 0}$, with $L \leq U$, are the *bounds* on the duration of b. An intermediate condition (IC) is a tuple $c = \langle \tau^\vdash, \tau^\dashv, \text{cond}(c) \rangle$ where $\tau^\vdash$ and $\tau^\dashv$ are the *relative times* denoting the start and end of when c has to hold, and $\text{cond}(c)$ is a consistent set of literals. If a literal $v, \neg w$ is in $\text{cond}(c)$ we say that c *needs* v or $\neg w$. An intermediate effect (IE) is a tuple $e = \langle \tau, \text{eff}(e) \rangle$ where τ is a relative time denoting when e is applied and $\text{eff}(e)$ is a consistent set of literals. If v or $\neg v \in \text{eff}(e)$ we say that e *adds* or *deletes* v, respectively, or, in general, that e *assigns* v. A *relative time* is either *action-relative* or *plan-relative*. An action-relative time is of the form $\text{START} + k$ or $\text{END} - k$ where START and END are *anchors* for the starting or ending

[1]There exists several ways to measure a dispatchment cost (Lamorgese and Mannino 2015), we chose the simplest one.

time, respectively, of the durative action containing the IC/IE, and $k \in \mathbb{Q}^{\geq 0}$ is the *offset* from the anchor. A plan-relative time is either ALPHA $+ k$ or OMEGA $- k$, with $k \in \mathbb{Q}^{>0}$, where ALPHA is the plan's first executed action start time (usually 0) and OMEGA is the plan's last action's end time. Depending on whether the relative times are action-relative or plan-relative we categorize the ICEs into *action/plan-ICs/IEs*, respectively. A *snap* or *instantaneous action* is a durative action with $L = U = 0$ and where all the ICEs in $\text{icond}(b) \cup \text{ieff}(b)$ have START as anchor in the relative time. To simplify notation, we refer to snap actions as a pair $a = \langle \text{pre}(a), \text{eff}(a) \rangle$ instead of the longer form for a durative action $\langle \{\langle \text{START}, \text{START}, \text{pre}(a) \rangle\}, \{\langle \text{START}, \text{eff}(a) \rangle\}, [0, 0] \rangle$. Let $b = \langle \text{icond}(b), \text{ieff}(b), [L, U] \rangle$ be a durative action. If $L = U$, we say that b *is well-orderable*, i.e., we can order each IC $\langle \tau^\vdash, \tau^\dashv, \text{cond}(c) \rangle \in \text{icond}(b)$ and each IE $\langle \tau, \text{eff}(e) \rangle \in \text{ieff}(b)$ of b accordingly to $\tau^\vdash[0, L]$ and $\tau[0, L]$ (Eq. 2), producing a sequence AICESEQ(b).

Finally, in the planning task Π: I is the *initial state*, where a *state* is a total assignment of the variables in V to $\{\top, \bot\}$; G is a set of conditions called *goals* or *goal conditions*; C and E are sets of plan-ICs and plan-IEs, respectively.

We say that (i) *two* IE*s are in mutex* if both assign a variable v, (ii) *an* IE e *and an* IC c *are in mutex* if e assigns a variable v and c needs either v or $\neg v$, and (iii) *a state* s *satisfies a set of literals* $L \subset \text{lit}(V)$, denoted with $s \models L$, if for each $v, \neg w \in L$ we have $s(v) = \top$ and $s(w) = \bot$, respectively. Let $e = \langle \tau, \text{eff}(e) \rangle$ be an IE. The *state resulting from applying* e *in* s is the state $s' = res(s, e)$ such that for each $v \in V$ we have $s'(v) = \top$ if $v \in \text{eff}(e)$, $s'(v) = \bot$ if $\neg v \in \text{eff}(e)$, and $s'(v) = s(v)$ otherwise. A *timed durative action* is a tuple $\langle t, b, d \rangle$ with $t \in \mathbb{Q}^{>0}$ being the *absolute time* in which the durative action $b = \langle \text{icond}(b), \text{ieff}(b), [L, U] \rangle$ is executed, lasting a duration $d \in [L, U]$. A *temporal plan* π for Π is a finite set of timed durative actions. The *make-span* of π is $\text{ms}(\pi) = \max(\{t + d \mid \langle t, b, d \rangle \in \pi\})$. The *absolute* IC*s of* π is the set $\text{icond}(\pi)$ containing (i) for each time durative action $\langle t, b, d \rangle \in \pi$ and for each IC $\langle \tau^\vdash, \tau^\dashv, \text{cond}(c) \rangle \in \text{icond}(b)$ the *absolute* IC $\langle \tau^\vdash[t, t+d], \tau^\dashv[t, t+d], c \rangle$ and (ii) for each $\langle \tau^\vdash, \tau^\dashv, c \rangle \in C$ the absolute IC $\langle \tau^\vdash[0, \text{ms}(\pi)], \tau^\dashv[0, \text{ms}(\pi)] \rangle$ where

$$\tau[a, b] = \begin{cases} a + k & \text{if } \tau = \text{START} + k \text{ or } \tau = \text{ALPHA} + k, \\ b - k & \text{if } \tau = \text{END} - k \text{ or } \tau = \text{OMEGA} - k. \end{cases} \quad (2)$$

Intuitively, the sets are called "absolute" since the time is expressed as absolute (i.e., in $\mathbb{Q}^{>0}$) instead of relative (e.g., START $+ k$). Similarly, we can define the *absolute* IE*s of* π $\text{ieff}(\pi)$. Let $\text{ieff}(\pi) = \{\langle t_1, \text{eff}(e_1) \rangle, \ldots, \langle t_n, \text{eff}(e_n) \rangle\}$. The sequence $\text{ieff}(\pi)$ applied from s_0 induces a sequence of *timed states* $s_0(\pi) = \langle 0, s_0 \rangle; \langle t_1, s_1 \rangle \ldots; \langle t_n, s_n \rangle$ where $s_i = res(s_{i-1}, e_i)$ for $i \in [1, n]$. We say that π is *valid* iff

1. *initial and goal condition*: $s_0 = I$ and $s_m \models G$.
2. *intermediate conditions:* for each $c = \langle t^\vdash, t^\dashv, \text{cond}(c) \rangle \in \text{icond}(\pi)$ (i) if $t^\vdash = t_0$, then $s_0 \models \text{cond}(c)$, (ii) for each $\langle t_i, s_i \rangle, \langle t_{i+1}, s_{i+1} \rangle$ in $s_0(\pi)$ s.t. either $t_i < t^\vdash \leq t_{i+1}$ or $t_i < t^\dashv \leq t_{i+1}$ then $s_i \models \text{cond}(c)$, (iii) if $t_n < t^\dashv$, then $s_n \models \text{cond}(c)$.
3. *no self-overlapping:* for each $\langle t, b, d \rangle, \langle t', b, d' \rangle$ in π, i.e., the application of the same durative action b at different times and with (possibly) different durations, it holds that either $t' + d' \leq t$ or $t' \geq t + d$,
4. ϵ*-separation:* let $\epsilon \in \mathbb{Q}^{\geq 0}$ and let $e = \langle t, \text{eff}(e) \rangle, e' = \langle t', \text{eff}(e') \rangle \in \text{ieff}(\pi)$ be two distinct IEs in π. If e and e' are in mutex then $|t - t'| \geq \epsilon$.

We considered the standard notion of validity, where the no self-overlapping property ensures that deciding plan-existence for Π is PSPACE-complete (Gigante et al. 2022b).

2.3 Symbolic Pattern Planning

Let $\Pi = \langle V, A, I, G, C, E \rangle$ be a planning task. A *pattern* $\prec = h_1; \ldots; h_k$, with length $k \geq 0$, is a finite sequence of *happenings* of Π. A happening h is a tuple $\langle \text{cond}(h), \text{eff}(h) \rangle$ where $\text{cond}(h)$ and $\text{eff}(h)$ are (possibly empty) sets of conditions and effects, respectively. Intuitively, we can model an IC $c = \langle \tau^\vdash, \tau^\dashv, \text{cond}(c) \rangle$ into a happening h with $\text{cond}(h) = \text{cond}(c)$ and $\text{eff}(h) = \emptyset$, and an IE $e = \langle \tau, \text{eff}(e) \rangle$ into a happening h' with $\text{cond}(h') = \emptyset$ and $\text{eff}(h') = \text{eff}(e)$. For an action $b \in A$, we denote by $\text{happ}(b)$ the happenings computed from $\text{icond}(b)$ and $\text{ieff}(b)$ as above. A pattern $\prec = h_1; \ldots; h_k$ thus intuitively represents the expected interleave of plan/action-ICs and plan/action-IEs. We say that a pattern is (i) *complete* if it contains, as happenings, all the ICs and IEs of Π, and (ii) *simple* if it contains, as happenings, an IC or an IE of Π at most once.

In (Cardellini and Giunchiglia 2026), an encoding was presented for a Temporal Planning Task[2] with ICEs. Intuitively, let Π be a planning task and $\prec$ be a pattern of Π. The *pattern-encoding* of Π using the pattern $\prec$ is the formula $\Pi^\prec$

$$\mathcal{I}(\mathcal{X}) \wedge \mathcal{S}^\prec(\mathcal{X}, \mathcal{H}^\prec, \mathcal{X}') \wedge \mathcal{T}^\prec(\mathcal{X}, \mathcal{H}^\prec, \mathcal{C}^\prec) \wedge \mathcal{G}(\mathcal{X}'), \quad (3)$$

where (i) $\mathcal{X} = V$ are the *current state variables*, (ii) $\mathcal{X}'$ are the *next state variables*, i.e., a copy of $\mathcal{X}$, (iii) $\mathcal{H}^\prec$ contains a variable with domain in $\{0, 1\}$ for each happening h_i in $\prec$, denoting whether h_i is applied, (iv) $\mathcal{C}^\prec$ contains the *clock* information, i.e., the timings of each happening h_i in $\prec$, (v) $\mathcal{I}(\mathcal{X})$ is the *initial state formula* over the variables in $\mathcal{X}$, (vi) $\mathcal{S}^\prec(\mathcal{X}, \mathcal{H}^\prec, \mathcal{X}')$ is the *causal transition relation*, characterizing the effects of the applications of IEs and the respect of ICs, (vii) $\mathcal{T}^\prec(\mathcal{X}, \mathcal{H}^\prec, \mathcal{C}^\prec)$ is the *temporal transition relation*, characterizing the temporal relations between ICEs, self-overlapping of actions and ϵ-separation, and, finally, $(viii)$ $\mathcal{G}(\mathcal{X}')$ is the *goal formula* constructed on $\mathcal{X}'$.

The SPP approaches starts constructing a pattern $\prec_I$ on the initial state I employing an Asymptotic Relaxed Planning Graph (ARPG) (Scala et al. 2016). A model for $\Pi^\prec$, i.e., a mapping of all the variables in $\mathcal{X}, \mathcal{X}', \mathcal{H}^\prec, \mathcal{C}^\prec$ to their respective domains, is searched employing a SMT solver (e.g., Z3 (De Moura and Bjørner 2008)) having $\prec = \prec_I$. If no model is found, the pattern is extended concatenating again $\prec$ to $\prec_I$, i.e., $\prec = \prec; \prec_I$, until a model is found. Let $\prec = h_1; \ldots; h_k$ be a pattern. In a model of $\Pi^\prec$, the temporal order between two happenings h_i, h_j with $i < j \leq k$ has to be respected only if h_i and h_j are in mutex. Let $\prec^n$ be the pattern obtained concatenating $\prec$ for $n \geq 1$ times. We say

[2] Cardellini and Giunchiglia (2026) deal also with numeric variables. In this paper, we consider a subset of their approach.

that $\Pi^{\prec}$ is (i) *correct* if any model of $\Pi^{\prec}$ corresponds to a valid plan of Π, (ii) *complete* if any valid plan corresponds to a model of $\Pi^{\prec^n}$ for some $n \geq 0$, and (iii) k*-complete* if any valid plan corresponds to a model of $\Pi^{\prec^k}$.

Theorem 1 (Cardellini and Giunchiglia 2026)**.** *Let* Π *be a planning task and* $\prec$ *be a complete pattern.* $\Pi^{\prec}$ *is correct and complete.*

Choosing the right $\prec$ is crucial for the efficiency of the approach, since the number of variables is proportional to the length of $\prec$ for which $\Pi^{\prec}$ is satisfiable.

3 In-Station Train Dispatching as Planning

Let $X = \langle Z, T, \eta, \sigma_0, F, o\rangle$ be an InSTraDi task with (i) station $Z = \langle S, B, P, E^+, E^-, R\rangle$, (ii) timetable $\eta = \langle \text{arrive}, \text{leave}\rangle$, (iii) initial state $\sigma_0 = \langle \text{lock}, \text{reserve}, \text{on}, \text{occupy}, \text{stopped}\rangle$, (iv) forecast $F = \langle f_M, f_S, f_D, f_E\rangle$, and (v) $o = \langle o^+, o^-\rangle$. To model the InSTraDi task X we define the planning task $\Pi_X = \langle V, A, I, G, C, E\rangle$, where:

The variables in V are

$locked(b)$: bit b is locked by a train,
$inFront(\tau, s)$: train τ is in front of signal s,
$green(\tau, s)$: signal s is green for train τ,
$stopping(\tau)$: train τ is stopping,
$occupies(\tau, p)$: train τ occupies platform p,
$stopped(\tau)$: train τ has stopped at a platform,
$stoppedAt(\tau, p)$: train τ has stopped at platform p,
$entered(\tau)$: train τ has entered the station,
$left(\tau)$: train τ has left the station.

In the initial state I, these variable are true

$$\{stoppedAt(\tau, p), stopped(\tau) \mid \tau \in T, p = \text{stopped}(\tau)\}$$
$$\cup \{inFront(\tau, \text{orig}(\tau)) \mid \tau \in T, \text{occupy}(\tau) \in P\}$$
$$\cup \{locked(b) \mid \tau \in T, b \in \text{lock}(\tau)\}$$
$$\cup \{green(\tau, s) \mid \tau \in T, \text{leave}(\tau) \leq 0, s \in \text{stops}(\tau)\},$$
$$\cup \{green(\tau, s) \mid \tau \in T, \text{obj}(\tau) = \text{T}, s \in \text{stops}(\tau)\},$$

i.e., the (idle) situation of the station in σ_0 is reported in I, the timetable of trains which should have already departed is accounted. All other variables are false.

The goal G is the set of conditions.

$$\{left(\tau) \mid \tau \in T, \text{obj}(\tau) \in \{\text{S}, \text{T}, \text{O}\}\} \cup$$
$$\{stopped(\tau) \mid \tau \in T, \text{obj}(\tau) \in \{\text{S}, \text{D}\}\}$$

i.e, trains must respect their objectives.

The set C is empty.

The set E contains

$$\{\texttt{arrival}(\tau) \mid \tau \in T, f_E(\tau) > 0, \text{obj}(\tau) \in \{\text{S}, \text{T}, \text{D}\}\} \cup$$
$$\{\texttt{timetable}(\tau) \mid \tau \in T, \text{leave}(\tau) > 0, \text{obj}(\tau) \in \{\text{S}, \text{O}\}\},$$

where

$$\texttt{arrival}(\tau) = \langle \text{ALPHA} + f_E(\tau), \{inFront(\tau, \text{orig}(\tau)), green(\tau, s)\}\rangle \quad (4)$$

$$\texttt{timetable}(\tau) = \langle \text{ALPHA} + \text{leave}(\tau), \{green(\tau, s) \mid p \in \text{stops}(\tau), s \in S_p\}\rangle \quad (5)$$

model, respectively, the arrival of trains at an entry point and the departure from a platform, according to F and η.

The set of actions A contains the following actions:

$\texttt{enter}(\tau, r)$ models the entrance in the station of train τ from route r. Let $r = b_1; \ldots; b_m$, $s = \text{from}(r)$, and $s' = \text{to}(r)$. Let $k_i = \sum_{j=1}^{i} f_M(\tau, r, b_j)$ for $i \in [1, m]$ be b_i's liberation time. The action has duration $L = U = f_M(\tau, r)$,

$$\text{ICs}: \{\langle \text{START}, \text{START}, \{inFront(\tau, s), green(\tau, s), \neg locked(b_1), \ldots, \neg locked(b_m)\} \cup \{entered(\tau') \mid \tau' \in T_o^+(\tau)\}\rangle\},$$
$$\text{IEs}: \{\langle \text{START}, \{\neg inFront(\tau, s), entered(\tau), locked(b_1), \ldots, locked(b_m)\}\rangle,$$
$$\langle \text{START} + k_1, \{\neg locked(b_1)\}\rangle,$$
$$\ldots$$
$$\langle \text{START} + k_{m-1}, \{\neg locked(b_{m-1})\}\rangle,$$
$$\langle \text{END}, \{inFront(\tau, s')\}\rangle\}.$$

Before moving, τ is in front of the green signal s, no bit of route r is locked, and all trains preceding τ in o have entered. When the movement starts, s becomes red and all the bits of r are locked. While the train moves, the IEs of the action free the bits (except the last one). When the action has ended and the movement completed, the train is now in front of signal s'.

$\texttt{stop}(\tau, s)$ models the stop of train τ facing signal s of platform p. The action has duration $L = U = f_S(\tau, p)$ and

$$\text{ICs}: \{\langle \text{START}, \text{START}, \{inFront(\tau, s), \neg stopped(\tau), \neg occupies(\tau, p)\}\rangle\},$$
$$\text{IEs}: \{\langle \text{START}, \{occupies(\tau, p), stopping(\tau)\}\rangle,$$
$$\langle \text{END}, \{stopped(\tau), stoppedAt(\tau, p), \neg stopping(\tau), \neg occupies(\tau, p)\}\rangle\}.$$

$\texttt{depart}(\tau, r)$ models the departure of train τ into route $r = b_1; \ldots; b_m$ from a platform p such that $s = \text{from}(r) \in S_p$. The action has duration $L = U = f_M(\tau, r)$ and

$$\text{ICs}: \{\langle \text{START}, \text{START}, \{inFront(\tau, s), green(\tau, s), stoppedAt(\tau, p), \neg locked(b_1), \ldots, \neg locked(b_m)\}\rangle\},$$
$$\text{IEs}: \{\langle \text{START}, \{\neg inFront(\tau, s), locked(b_1), \ldots, locked(b_m)\}\rangle,$$
$$\langle \text{START} + f_D(\tau, p), \{\neg locked(b) \mid b \in \text{bits}(p)\}\rangle,$$
$$\langle \text{START} + k_1, \{\neg locked(b_1)\}\rangle,$$
$$\ldots$$
$$\langle \text{START} + k_{m-1}, \{\neg locked(b_{m-1})\}\rangle,$$
$$\langle \text{END}, \{inFront(\tau, s')\}\rangle\}.$$

Intuitively, the train departs only if it is in front of a stop signal, has stopped, the timetable time has passed (i.e., the signal is green) and all the bits in r are locked. The action mimics the `enter` action in the IEs, adding an IE after $f_D(\tau, p)$ to free all the track circuits of the platform.

$\texttt{exit}(\tau, r)$ models the exit from the station of train τ through route r. Let $s = \text{to}(r)$ and $r = c_1; \ldots; c_m$. The snap action is modelled as $\langle \text{pre}(a), \text{eff}(a)\rangle$ where

$$\text{pre}(a) = \{inFront(\tau, s)\} \cup \{left(\tau') \mid \tau' \in T_o^-(\tau)\}$$
$$\text{eff}(a) = \{left(\tau), \neg inFront(\tau, s), \neg locked(c_m)\}.$$

Intuitively, τ exits if all the other trains before it, according to o^-, have left. Then, the last bit of r is unlocked.

Notice that all the actions are well-orderable, having $L = U$.

Clearly, one could ground all the actions presented above for all trains, routes, signals and platforms. However, we perform the following preliminary steps, which do not alter the approach: (i) for a train τ, we consider only routes in

R_τ, (ii) we ground the actions $\texttt{enter}(\tau, r)$ and $\texttt{stop}(\tau, r)$ (resp. $\texttt{depart}(\tau, r)$ and $\texttt{exit}(\tau, r)$) only for trains with $\text{obj}(\tau) \in \{\text{S,T,D}\}$ (resp. $\text{obj}(\tau) \in \{\text{S,T,O}\}$), (iii) for actions $\texttt{stop}(\tau, s)$ we consider only signals s of a platform in $\text{stops}(\tau)$, (iv) for action $\texttt{depart}(\tau, r)$ we consider only routes r originating in a signal of a platform, (v) for action $\texttt{exit}(\tau, r)$ we consider only routes where $\text{to}(r) \in E^-$. We ground the actions $\texttt{stop}(\tau, s)$ and $\texttt{depart}(\tau, r)$ also for a train with $\text{obj}(\tau) = \text{T}$, since, even if the train shouldn't stop, it might be helpful to stop it (without boarding/alighting passengers), allowing other trains to overtake it.

Example 1. *Suppose we are in the setting presented in Fig. 1 with trains* `r`*,* `g` *and* `b`*. The* `r` *train* (i) *occupies platform* `II` *and bits* `114`*,* `115` *and* `116`*, facing signal* `22`*,* (ii) *departs, according to the timetable, in at least* $75s$*, and* (iii) *must exit from* `02`*. The* `b` *(resp.* `g`*) train* (i) *is arriving in signal* `03` *(resp.* `01`*) in* $5s$ *(resp.* $30s$*),* (ii) *it must stop at some platform and departure in at least* $70s$ *(resp.* $200s$*), and* (iii) *exit from* `02` *(resp.* `04`*). The* `b` *train must exit before the* `r` *train. Then the initial condition* I *has the following variables true*

$$\{\textit{stoppedAt}(\texttt{r}, \texttt{II}), \textit{stopped}(\texttt{r}), \textit{inFront}(\texttt{r}, \texttt{22}),$$
$$\textit{occupied}(\texttt{114}), \textit{occupied}(\texttt{115}), \textit{occupied}(\texttt{116})\}.$$

The goal G *is*

$$\{\textit{stopped}(\texttt{g}), \textit{stopped}(\texttt{b}), \textit{left}(\texttt{r}), \textit{left}(\texttt{g}), \textit{left}(\texttt{b})\}.$$

Suppose that, in the forecast, f_M *returns* 5 *for all the bits and* f_S *and* f_D *return* 10 *for all platforms. A possible valid plan* π *for* Π*, assuming* $\epsilon = 0.1$*, is*

$$\{\langle 5.1, \texttt{enter}(\texttt{b}, \texttt{03-23}), 30\rangle, \langle 30.1, \texttt{enter}(\texttt{g}, \texttt{01-41}), 40\rangle,$$
$$\langle 35.2, \texttt{stop}(\texttt{b}, \texttt{23}), 10\rangle, \langle 70.2, \texttt{stop}(\texttt{g}, \texttt{41}), 10\rangle,$$
$$\langle 70.1, \texttt{depart}(\texttt{b}, \texttt{21-02}), 15\rangle, \langle 85.2, \texttt{exit}(\texttt{b}, \texttt{21-02}), 0\rangle,$$
$$\langle 85.3, \texttt{depart}(\texttt{r}, \texttt{22-02}), 25\rangle, \langle 110.4, \texttt{exit}(\texttt{r}, \texttt{22-02}), 0\rangle,$$
$$\langle 200.1, \texttt{depart}(\texttt{g}, \texttt{41-04}), 25\rangle, \langle 225.2, \texttt{exit}(\texttt{g}, \texttt{41-04}), 0\rangle\}$$

In the plan, `b` *(resp.* `g`*) enters as soon it arrives at the entry point* `03` *(resp.* `01`*) at* $5s$ *(resp.* $35s$*), through route* `03-23` *(resp.* `01-41`*) and then it stops in platform* `III` *(resp.* `I`*). Since* `b` *must exit before* `r`*, it is the first to depart from platform* `I`*. After,* `b` *has exited,* `r` *can start to move towards the exit even if its departure time has passed. Train* `g` *leaves in time at* $200s$.

Let π be a plan for Π_X. We can map π into a dispatchment D for X by mapping each timed durative action $\langle t, b\rangle$ in π to a set of timed commands.

4 Patterns for In-Station Train Dispatching

Let X be an INSTRADI task. In this section, we explore how to compute a pattern $\prec$ for the planning task Π_X such that, all plans of Π_X corresponds to models of the encoding $\Pi_X^{\prec}$. First, we recognize some properties of the plans of Π_X.

Lemma 1. *Let* $\pi = \{\langle t_1, a_1, d_1\rangle, \ldots, \langle t_n, a_n, d_n\rangle\}$ *be a plan of* Π_X*. Then each* a_i *for* $i \in [1, n]$ *is distinct.*

Proof. Actions $\texttt{enter}(\tau, r)$, $\texttt{depart}(\tau, r)$ and $\texttt{exit}(\tau, r)$ cannot be repeated because of the acyclicity of the station graph and of the $\textit{inFront}(\tau, \text{from}(r))$ being deleted by their effects. A $\texttt{stop}(\tau, s)$ action cannot be repeated due to the $\textit{stopped}(\tau)$ predicate being permanently set to true. □

Lemma 2. *Let* X *be an* INSTRADI *task with only one train* τ*. If* (i) $\text{obj}(\tau) \in \{\text{S, T}\}$*, the plan for* Π_X *is* $\pi_{\text{S,T}}$*,* (ii) *if* $\text{obj}(\tau) \in \{\text{D}\}$ *is* π_{D}*, or* (iii) *if* $\text{obj}(\tau) \in \{\text{O}\}$ *is* π_{O}*, with*

$$\pi_{\text{D}} = \{\langle t_1, \textit{enter}(\tau, r), d_1\rangle, \langle t_2, \textit{stop}(\tau, \text{to}(r)), d_2\rangle\}$$
$$\pi_{\text{O}} = \{\langle t_3, \textit{depart}(\tau, r'), d_3\rangle, \langle t_4, \textit{exit}(\tau, r'), 0\rangle\}$$

and $\pi_{\text{S,T}} = \pi_{\text{D}} \cup \pi_{\text{O}}$ *with* $t_1 < t_2 \leq t_3 < t_4$ *for some* $r, r' \in R_\tau$*.*

Proof. An IC of $\texttt{enter}(\tau, r)$ needs $\textit{green}(\tau, \text{from}(r))$, added by the IE $\texttt{arrival}(\tau)$ (Eq. 4). An IC of $\texttt{stop}(\tau, \text{to}(r))$ needs $\textit{inFront}(\tau, \text{to}(r))$, added by an IE of $\texttt{enter}(\tau, r)$. An IC of $\texttt{depart}(\tau, r)$ needs $\textit{stoppedAt}(\tau, p)$ with p a platform where r terminates, added by an IE of $\texttt{stop}(\tau, p)$. An IC of $\texttt{depart}(\tau, r')$ needs $\textit{stoppedAt}(\tau, p)$, added by an IE of $\texttt{stop}(\tau, \text{to}(r))$. An IC of $\texttt{exit}(\tau, r)$ needs $\textit{inFront}(\tau, \text{to}(r'))$, added by an IE of $\texttt{depart}(\tau, r')$. Actions cannot be repeated (Lem. 1). □

Let τ be a train. We can construct a *pattern* $\prec_\tau$ *for* τ from the concatenation of (if present in $A \cup E$) (i) the IE $\texttt{arrival}(\tau)$, (ii) AICESEQ($\texttt{enter}(\tau, r)$) for each r originating in $\text{orig}(\tau)$, (iii) AICESEQ($\texttt{stop}(\tau, s)$) for each signal s delimiting a platform, (iv) the IE $\texttt{timetable}(\tau)$, (v) AICESEQ($\texttt{depart}(\tau, r)$) for each r originating from a signal of a platform, and (vi) AICESEQ($\texttt{exit}(\tau, r)$) for each r terminating in $\text{dest}(\tau)$.

Theorem 2. *Let* $T = \{\tau_1, \ldots, \tau_p\}$ *be the set of trains and let* $T_{\text{S,T}} \subseteq T$ *and* $T_{\text{O,D}} \subseteq T$ *be the trains with objective in* $\{\text{S, T}\}$ *and* $\{\text{O, D}\}$*, respectively. Let* $\prec = (\prec_{\tau_1}; \ldots; \prec_{\tau_p})^m$ *where* $m = \max(2|T_{\text{S,T}}| - 1, 0) + |T_{\text{O,D}}|$*. Then* $\Pi_X^{\prec}$ *is* 1*-complete.*

Proof. We first prove that with $m' = 2|T_{\text{S,T}}| + |T_{\text{O,D}}|$ and $\prec' = (\prec_{\tau_1}; \ldots; \prec_{\tau_p})^{m'}$ then $\Pi_X^{\prec'}$ is 1-complete. We have to consider all orders between all $p = |T_{\text{S,T}}| + |T_{\text{O,D}}|$ trains. Consider a valid plan $\pi = \pi_1 \cup \cdots \cup \pi_p$, with π_i being the plan relative to τ_i (Lem. 2). For each valid plan π, there exists a model μ of $\Pi_X^{\prec'}$ where, at each of the m' concatenations of $\prec_{\tau_1}; \ldots; \prec_{\tau_p}$, μ selects, for some train τ_i, either (i) all the happenings of the actions `enter` and `stop` (π_i is like π_{D} in Lem. 2), or (ii) the actions `depart` and `exit` (π_i is like π_{O} in Lem. 2). For trains in $T_{\text{S,T}}$ it has to choose both (π_i is like $\pi_{\text{S,T}} = \pi_{\text{O}} \cup \pi_{\text{D}}$ in Lem. 2) at two different concatenations, while, for trains in $T_{\text{O,D}}$, it has to choose either (i) or (ii) at a single concatenation. Due to Lem. 1, we have to select each action at most once. Thus $\Pi^{\prec'}$ is 1-complete. Regarding $\Pi^{\prec}$, the $2|T_{\text{S,T}}|$ concatenations are required only if we select, for each concatenation, the happenings in (i) before the happenings in (ii) for all trains in $T_{\text{S,T}}$. However, this is not a valid plan (Lem. 2). Thus, we stop at $2|T_{\text{S,T}}| - 1$ concatenations (if $|T_{\text{S,T}}| > 0$) and $\Pi^{\prec}$ is 1-complete. □

Thm. 2 presents a way to construct $\prec$ considering *all* possible orders between trains' movements. However, we know that the train order o forbids some orders. In the following, we show a construction method for $\prec$ that considers o.

Let X be an INSTRADI task and Π_X be a planning task for X. To compute a pattern for Π_X we construct a *Happening Dependency Network* (HDN). An HDN is a graph $N =$

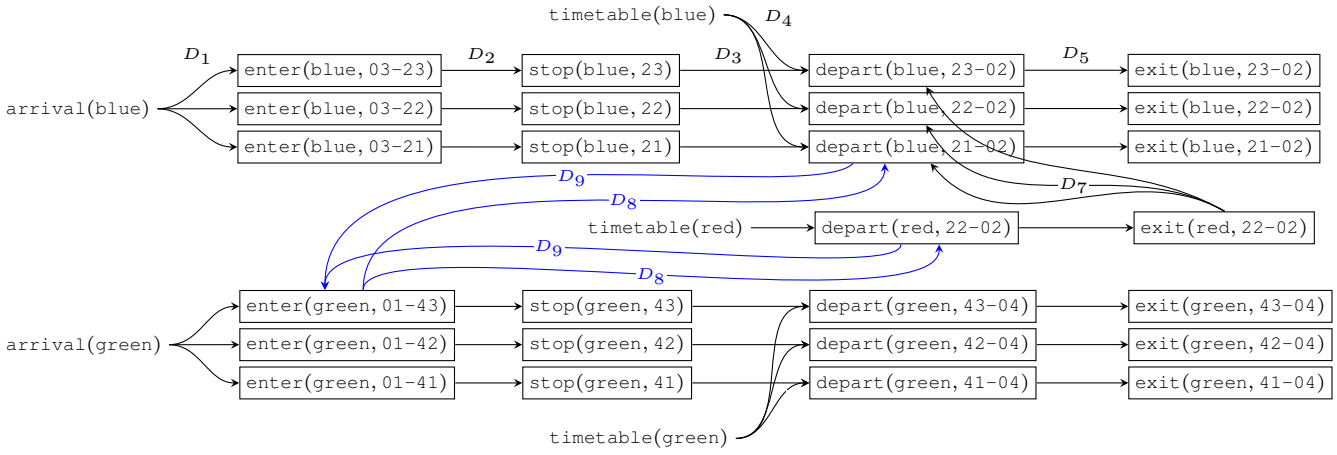


Figure 2: An example HDN N_X for the INSTRADI task of Fig. 1. Boxed nodes represent actions in A. Black edges are the dependencies D_1 to D_7. Blue edges are the dependencies D_8 to D_{11}. To avoid visual clutter, not all blue dependencies are drawn.

$\langle H, D\rangle$ where H are the nodes (or happenings) and $D \subseteq H \times H$ are the edges (or *dependencies*). For the INSTRADI task X we compute $N_X = \langle H, D\rangle$ such that $H = A \cup E$, i.e., the durative actions and the IEs of Eqs. 4 and 5. For each train τ and each $r, r' \in R_\tau$ originating and terminating in $\mathrm{orig}(\tau)$ and $\mathrm{dest}(\tau)$, respectively, we have[3] in D

D_1 $\langle \texttt{arrival}(\tau), \texttt{enter}(\tau, r)\rangle$,

D_2 $\langle \texttt{enter}(\tau, r), \texttt{stop}(\tau, \mathrm{to}(r))\rangle$,

D_3 $\langle \texttt{stop}(\tau, \mathrm{from}(r')), \texttt{depart}(\tau, r')\rangle$,

D_4 $\langle \texttt{timetable}(\tau), \texttt{depart}(\tau, r')\rangle$,

D_5 $\langle \texttt{depart}(\tau, r'), \texttt{exit}(\tau, r')\rangle$.

Intuitively, for each train τ we model that (i) the train arrives and moves towards the platform (D_1), stops (D_2), departs (D_3), and exits (D_5), and (ii) that the departure always come after the train's timetable departure (D_4).

Let $o = \langle o^+, o^-\rangle$ be the train order in X. For each pair of trains τ, τ' we have in D,

D_6 $\langle \texttt{enter}(\tau, r), \texttt{enter}(\tau', r')\rangle$ for each $r \in R_\tau$ and $r' \in R_{\tau'}$ both originating in an entry point, if $o^+(\tau, \tau') < 0$,

D_7 $\langle \texttt{exit}(\tau, r), \texttt{depart}(\tau', r')\rangle$ for each $r \in R_\tau$ and $r' \in R_{\tau'}$ both terminating in an exit point, if $o^-(\tau, \tau') < 0$.

Intuitively imposing an order between trains according the entry order o^+ (D_6) and exit order o^- (D_7).

The edges in N_X imposed until now with $D_1, \ldots, D_7$, semantically impose that if $\langle h_i, h_j\rangle \in D$, then h_i *must* happen before h_j. The graph N_X is thus a Direct Acyclic Graph (DAG), due to (i) the acyclicity of the station graph ($D_1, \ldots, D_5$), and (ii) for the partial order of the functions in o (D_6, D_7). Fig. 2 presents the HDN $N_X = \langle H, D\rangle$ constructed until now (black edges). A pattern $\prec = h_1; \ldots; h_k$ can be constructed from N_X by *linearization*, i.e., by finding a *topological order* between all the nodes in H and by substituting each node $b \in A$ representing a durative action with the sequence of happenings AICESEQ(b).

[3]An edge $\langle h_i, h_j\rangle$ is created only if both $h_i, h_j \in H$.

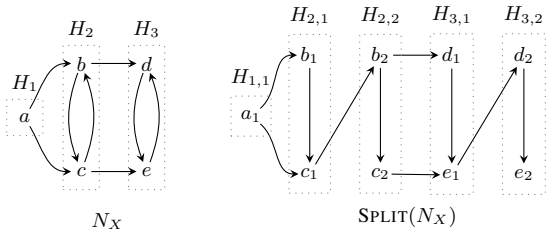


Figure 3: Example of how a HDN N_X with 5 nodes and 3 SCCs is split in SPLIT(N_X). The SCC H_2 with 2 nodes is split in (suppose) $m_2 = 2$ copies $H_{2,1}$ and $H_{2,2}$. The order chosen $<_2$ has $b <_2 c$ and thus (b_1, c_1),(b_2, c_2),$(c_1, b_2) \in$ SPLIT(D). The same holds for H_3. Since $(b, d), (c, e) \in D$ we have $(b_2, d_1), (c_2, e_1) \in$ SPLIT(D).

Consider, in Fig. 1, where `r` exits through action `depart(r, 22-02)` and `g` enters from any route that originate in `01` (e.g., `01-21`). Any pattern $\prec$ obtained by the HDN in Fig. 2 (black edges) would cause, e.g., either `depart(r, 22-02)` before `enter(g, 01-21)` or viceversa. Since the two actions have IEs in mutex (i.e., the one locking bits `111`, `112` and `113`), any model in $\Pi_X^{\prec}$ would have to respect the order provided by $\prec$, thus pruning valid plans (i.e., $\Pi_X^{\prec}$ is not 1-complete). For this reason, for each pair of distinct trains τ, τ' and routes $r \in R_\tau$ and $r' \in R_{\tau'}$, such that r and r' are intersecting, we add in D

D_8 $\langle \texttt{enter}(\tau, r), \texttt{depart}(\tau', r')\rangle$,

D_9 $\langle \texttt{depart}(\tau, r), \texttt{enter}(\tau', r')\rangle$

D_{10} $\langle \texttt{enter}(\tau, r), \texttt{enter}(\tau', r')\rangle$ if r and r' do not originate in the same entry point, and

D_{11} $\langle \texttt{depart}(\tau, r), \texttt{depart}(\tau', r')\rangle$ if either r and r' do not terminate in the same exit point or $o^-(\tau, \tau') = 0$.

Clearly, due to the symmetry of intersection of routes, the dependencies in $D_8, \ldots, D_{11}$ cause two edges for each distinct pair of routes, like the blue edges depicted in Fig. 2, causing $N_X = \langle H, D\rangle$ to become cyclic. For this reason, we construct the HDN SPLIT(N_X) = $\langle$SPLIT(H), SPLIT(D)$\rangle$:

1. we compute the Strongly Connected Components (SCCs) of N_X, i.e., a partition $H_1, \ldots, H_n$ of the nodes in H,

2. for each SCC H_p, with $p \in [1, n]$,
   (a) we select an order $<_p$ among the happenings in H_p s.t., among all possible trains, routes and signals, we have `arrival` $<_p$ `enter` $<_p$ `stop` $<_p$ `timetable` $<_p$ `depart` $<_p$ `exit`,
   (b) we denote by $T^p_{\text{S,T}}$ and $T^p_{\text{O,D}}$ the set of trains whose IEs or actions appear in H_p and with objective in $\{\text{S}, \text{T}\}$ and $\{\text{O}, \text{D}\}$, respectively,
   (c) we add to SPLIT(H) the nodes in SPLIT$(H_p) = H_{p,1} \cup \cdots \cup H_{p,m_p}$ having $m_p = \max(2|T^p_{\text{S,T}}| - 1, 0) + |T^p_{\text{O,D}}|$ copies of the happenings in H_p
   (d) for each $\langle h, h' \rangle \in D$ with $h, h' \in H_p$
      i. if $h <_p h'$ we add an edge $\langle h_{p,q}, h'_{p,q} \rangle$ to SPLIT(D), for each $q \in [1, m_p]$, with $h_{p,q}$ and $h'_{p,q}$ being the copy of h and h' in $H_{p,q}$,
      ii. if $h' <_p h$ we add an edge $\langle h_{p,q}, h'_{p,q+1} \rangle$ to SPLIT(D) for each $q \in [1, m_p - 1]$, with $h_{p,q}$ and $h'_{p,q+1}$ being the copy of h and h' in $H_{p,q}$ and $H_{p,q+1}$.
3. for each edge $\langle h_p, h_r \rangle \in D$ with h_p in the SCC H_p of size m_p and $h_r \in H_r$ of size m_r, with $r \neq p$, i.e., two different SCCs, we add an edge $\langle h_{p,m_p}, h_{r,1} \rangle$ in SPLIT(D).

An example of how SPLIT(N_X) is computed is shown in Fig. 3. SPLIT(N_X) is acyclic by construction, since each SCC in N_X is transformed into an acyclic graph.

Theorem 3. *Let N_X be a* HDN *for X. Let $\prec$ be the linearization of* SPLIT(N_X). *Then $\Pi^{\prec}_X$ is 1-complete.*

Proof. Let $N_X = \langle H, D \rangle$. Let π be a valid plan for Π_X. Let $t_\pi \colon A \cup E \mapsto \mathbb{Q}^{>0} \cup \{\infty\}$ be a function mapping each durative action A and each plan-IE in E (i.e., the nodes of N_X) to the time when they are scheduled to happen in π, with ∞ denoting that the action or IE is never applied. We denote by $N_\pi = \langle H_\pi, D_\pi \rangle$ the subgraph of N_X such that $H_\pi = \{h \mid h \in H, t_\pi(h) \neq \infty\}$ and $D_\pi = \{\langle h, h' \rangle \mid \langle h, h' \rangle \in D, h, h' \in H_\pi, t_\pi(h) < t_\pi(h')\}$, i.e., all the nodes which are present in π and all edges $\langle h, h' \rangle$ where h happens before h' in π. N_π still has the edges in D_1,...,D_5, due to Lem. 2 and all the edges in D_6 and D_7, due to ICs of the actions `enter` and `exit` concerning o^+ and o^-, respectively. N_π is acyclic, since, for each pair of edges $\langle h, h' \rangle, \langle h', h \rangle$ introduced by D_8,$\cdots$,D_{11}, we remove one depending on if $t_\pi(h) < t_\pi(h')$ or viceversa. Consider a linearization $\prec_\pi$ of N_π. By construction, $\Pi^{\prec_\pi}$ is satisfiable since it has a model corresponding to π. Now we prove that $\prec_\pi$ is a subsequence of any pattern $\prec$ obtained from linearizing SPLIT(N_X) (i.e., $\prec_\pi$ can be obtained from $\prec$ by removing some happenings). For each SCC H_p in N_X, we construct in SPLIT(N_X) a number m_p of copies $H_{p,1}, \ldots; H_{p,m_p}$. Due to the order $<_p$, the plan π can select, for some train, at each $H_{p,q}$ with $q \in [1, m_p]$, (i) the actions `enter` and `stop`, or (ii) the actions `depart` and `exit`. For trains in $T^p_{\text{S,T}}$ it has to choose both (i) and (ii) at two $H_{p,q}$ and $H_{p,q'}$ with $q \neq q'$, while, for trains in $T^p_{\text{O,D}}$, it has to choose either (i) or (ii) at some $H_{p,q}$ (Lem. 2). Since we make $m_p = \max(2|T^p_{\text{S,T}}| - 1, 0) + |T^p_{\text{O,D}}|$ copies of the happenings in H_p, all possible orders can be selected by π in SPLIT(H_p), i.e., the pattern $\prec_\pi$ is a substring of the pattern $\prec$. Thus if $\Pi^{\prec_\pi}$ has a model, also $\Pi^{\prec}$ has a model, since a model can simply not select all happenings which are in $\prec$ and not in $\prec_\pi$. Thus, since for each valid plan π the exists a model in $\Pi^{\prec}$, $\Pi^{\prec}$ is 1-complete.

□

We now have a way to compute a pattern $\prec$ from N_X that guarantees that $\Pi^{\prec}$ is 1-complete. However, consider the following scenario in Fig. 1: suppose that train `r` departs in one minute, while train `g` arrives in one hour. The HDN would still consider the dependencies in D_8 to D_{11} between the movements of `r` and `g` because the plan where `r` waits one hour to let `g` pass is indeed a valid plan, although, clearly not an optimal one. For this reason, we introduce a *linking threshold* $\Delta \in \mathbb{N}$ and we construct the HDN $N_{X,\Delta}$ which has all the dependencies from D_1 to D_7 together with the following new dependencies for each pair of trains τ, τ' and for each route $r, r' \in R$ we have in D

D_{12} $\langle \texttt{exit}(\tau, r), \texttt{enter}(\tau', r') \rangle$ if $f_E(\tau') - \text{leave}(\tau) > \Delta$,

D_{13} $\langle \texttt{enter}(\tau, r), \texttt{enter}(\tau', r') \rangle$ if $f_E(\tau') - f_E(\tau) > \Delta$,

D_{14} $\langle \texttt{enter}(\tau, r), \texttt{depart}(\tau', r') \rangle$ if $\text{leave}(\tau') - f_E(\tau) > \Delta$,

D_{15} $\langle \texttt{depart}(\tau, r), \texttt{depart}(\tau', r') \rangle$ if $\text{leave}(\tau') - \text{leave}(\tau) > \Delta$,

D_{16} the dependencies in D_8 only if $|f_E(\tau) - \text{leave}(\tau')| \leq \Delta$,

D_{17} the dependencies in D_9 only if $|\text{leave}(\tau) - f_E(\tau')| \leq \Delta$,

D_{18} the dependencies in D_{10} only if $|f_E(\tau) - f_E(\tau')| \leq \Delta$,

D_{19} the dependencies in D_{11} only if $|\text{leave}(\tau) - \text{leave}(\tau')| \leq \Delta$.

Theorem 4. *Let Δ be a linking threshold and let $N_{X,\Delta}$ be a* HDN *for X. Let $\prec$ be the linearization of* SIMPLIFY$(N_{X,\Delta})$. *Then $\Pi^{\prec}_X$ is <u>not</u> 1-complete.*

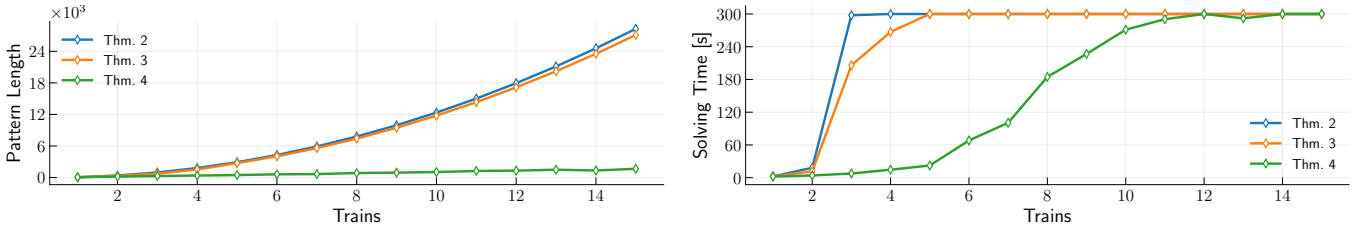
Proof. Clearly, not all solutions are accounted, since for trains where the linking threshold is not respected, the linearization imposes an order that has to be respected by all models, thus pruning valid plans. □

5 Experimental Analysis

We now empirically evaluate the performance of the proposed approaches. The analysis is conducted on the station depicted in Fig. 1. Although the station is synthetic, its modelling is inspired by a real station in the North-West of Italy. The structure has been validated by relevant stakeholders, namely the Italian Railway Network company and a major worldwide train signalling company.[4] This work is part of a broader research project aimed at introducing AI-based planning techniques into the railway domain and empirically assessing their benefits.

Performance is evaluated on scenarios with an increasing number of trains operating within the station. We generated 15 INSTRADI tasks, with the number of trains ranging from 1 to 15. Let $T = \{\tau_1, \ldots, \tau_n\}$ be a scenario with $n \geq 1$ trains. All trains are assigned objective $\text{obj}(\tau_i) = \text{S}$, since, as shown in Thm. 2, this objective induces the longest patterns and therefore represents the most challenging setting. For

[4]The company name is anonymized. Due to a NDA, we are not authorized to disclose the name and structure of the real station; however, the synthetic station data will be released upon acceptance



(a) Pattern Length

(b) Solving Time

Figure 4: Comparison on the pattern length (Fig. 4a) and the solving time in seconds (Fig. 4b) with respect to the number of trains in T.

each train τ_i, with $i \in [1, n]$, the forecasted arrival time $f_E(\tau_i)$ is sampled uniformly from the interval $[600 \times i - 180, 600 \times i + 180]$, corresponding to arrivals every 10 ± 3 minutes. The scheduled departure time $\text{leave}(\tau_i)$ is set to 8 ± 3 minutes after arrival. These timing distributions are derived from three months of operational data collected in 2025 at the real-world station inspiring Fig. 1, and were validated by the stakeholders. The relative ordering of trains is determined by arrival times for entry and by departure times for exit. For each distinct pair τ_i, τ_j, if $\text{orig}(\tau_i) \neq \text{orig}(\tau_j)$ then $o^+(\tau_i, \tau_j) = 0$; otherwise, if $f_E(\tau_i) < f_E(\tau_j)$ then $o^+(\tau_i, \tau_j) = -1$, and $o^+(\tau_i, \tau_j) = 1$ otherwise. The exit ordering $o^-(\tau_i, \tau_j)$ is defined analogously, based on $\text{dest}(\tau_i)$, $\text{dest}(\tau_j)$, $\text{leave}(\tau_i)$, and $\text{leave}(\tau_j)$. For Thm. 4, the linking threshold is set to $\Delta = 300$ seconds (i.e., 5 minutes). The forecast F is computed by averaging, for each train, the observed durations of movements, stops, and departures in the real station over the aforementioned three-month period.

We compare the three domain-dependent pattern-generation methods presented in Thm. 2, Thm. 3, and Thm. 4. We modified the solver PATTY (Cardellini, Giunchiglia, and Maratea 2026), the SoTA planner for SPP extended to handle ICEs (Cardellini and Giunchiglia 2026), by replacing its domain-independent pattern computation module—based on the ARPG (Scala et al. 2016)—with our domain-dependent techniques. For each generated INSTRADI task X, we construct the planning task Π_X and encode it in ANML (Smith, Frank, and Cushing 2008), the de-facto standard language for temporal planning with ICEs. Given Π_X and a domain-dependent pattern $\prec$ (computed according to one of the three theorems above), PATTY produces the SMT formula $\Pi_X^{\prec}$ (Eq. 3) in SMT-LIB format (Barrett, Fontaine, and Tinelli 2016), which is then solved using the SMT-solver `Z3` v4.13 (De Moura and Bjørner 2008). The optimization objective is encoded using the SMT-LIB command `minimize`. In particular, we encode the dispatching objective of Eq. 1 by replacing $\text{realLeave}(D, \tau)$ with the sum of the variables in $\mathcal{H}^{\prec}$ of $\Pi_X^{\prec}$ representing the time of the first IC of $\texttt{depart}(\tau, r)$, for each $r \in R_\tau$ originating from a platform. By construction of the encoding (Cardellini and Giunchiglia 2026), if an happening is not selected in the model its time is set to 0, so the sum effectively corresponds to the departure action chosen by the solver. The model returned by `Z3` is translated by PATTY into a plan, which is then converted into a dispatchment and validated as described in Sec. 2.1.

We didn't compare our approach with existing approaches from the literature, since several factors limit the feasibility and fairness of such an evaluation. First, the approach by Cardellini et al. (2021) (the only one existing employing planning) does not provide guarantees of optimality, as it follows a greedy, albeit effective, strategy that makes locally optimal decisions to quickly reach a solution. While efficient, this strategy may preclude globally optimal solutions, for instance in cases where delaying a train would improve the overall schedule. This is a known limitation of search-based planning approaches, where early decisions are difficult to revise. In contrast, our method guarantees that an optimal solution is included in the solution set and can be identified through optimization. Furthermore, no MILP/CP-based systems are publicly available, making direct comparison difficult. Existing studies are often tightly coupled with specific industrial use cases under NDAs, resulting in a proliferation of partially incompatible problem definitions and limited opportunities for performance assessment (Kloster et al. 2025).

Fig. 4 reports scalability with respect to the number of trains. For each problem size, we generated 10 instances and report averaged results. Fig. 4a shows the pattern length for the three approaches. As expected, Thm. 2 yields the longest patterns, since it replicates the base pattern $2m - 1$ times for an instance with m trains all having objective S. The approach of Thm. 3 produces slightly shorter patterns (approximately one thousand fewer happenings in the case with 15 trains), but the overall growth trend remains comparable. To illustrate, consider three trains $T = \{\tau_1, \tau_2, \tau_3\}$ where τ_1 and τ_3 share the same origin and destination, while τ_2 differs in both. In the HDN N_X, an order is imposed between the actions of τ_1 and τ_3. However, τ_2 has conflicting routes with both τ_1 and τ_3, inducing dependencies D_8–D_{11} between τ_1 and τ_2 and between τ_2 and τ_3. As a consequence, the resulting SCC in the HDN contains almost all actions of τ_1, τ_2, and τ_3, thus limiting its effectiveness. In contrast, the approach of Thm. 4 substantially reduces the pattern size. Given the arrival and departure distributions, most trains are temporally separated by more than the linking threshold Δ. Only a few pairs of temporally close trains require considering alternative orderings, resulting in smaller patterns.

Fig. 4b reports the total solving time, including: translation from X to Π_X, pattern computation, encoding of $\Pi_X^{\prec}$, `Z3` solving time, and validation of both the plan and the dispatchment. The reduction in pattern size directly translates into improved solving times, with the approach of Thm. 4 consistently outperforming the others. For all instances solved in the cut-off of 5 minutes, we verified that the three approaches produced solutions of identical (optimal) quality.

6 Conclusions and Future Work

In this paper, we formally introduced the InSTraDi problem and provided a temporal planning encoding with ICEs such that every valid plan for the planning task corresponds to a valid dispatchment for InSTraDi. We also presented a domain-independent method to compute a pattern within the SPP framework that guarantees 1-completeness, i.e., all valid plans—and therefore the optimal one—can be found using the computed pattern. Our experimental analysis indicates that the proposed techniques significantly reduce the pattern length, which in turn decreases the overall solving time while preserving optimality guarantees. In particular, the simplified construction enables the computation of provably optimal dispatchments more efficiently than previous approaches.

We plan to extend this work along two directions, both domain-dependent and independent. From a domain-dependent perspective, we aim to model larger and more complex stations. In such settings, train movements are not limited to entry–platform–exit sequences, but involve intermediate signals where trains can temporarily stop to allow overtaking maneuvers. Secondly, we intend to incorporate *shunting operations*, where trains with long dwell times are moved from platforms to dedicated parking areas within the station to free critical resources. Finally, we plan to extend our approach to support the DISPLIB library (Kloster et al. 2025). The DISPLIB initiative was introduced to address key limitations in the field, namely the lack of publicly available code and datasets, which hinders reproducibility and makes it difficult to fairly compare approaches. We are currently collaborating with our industrial partner to construct benchmarks in the DISPLIB format. From a domain-independent perspective, the pattern-construction method developed for InSTraDi can potentially be generalized to other problems, and more broadly to planning domains exhibiting similar structural properties. Our procedure relies on the assumption of Lem. 1, namely that each action cannot be executed more than once. We therefore plan to investigate additional domains satisfying this property and to design a general, domain-independent methodology for constructing the corresponding HDN, from which a complete pattern is derived.

AI Declaration

The authors have not employed any Generative AI tools.

Aknowledgments

Funded by the European Union under Grant agreement 101101973 (MOTIONAL Project). Views and opinion expressed herein are however those of the author(s) only and do not necessarily reflect those of the European Union or Europe's Rail Joint Undertaking. Neither the European Union nor the granting authorities can be held responsible for them. The project is supported by the Europe's Rail JE and its members. M. Cardellini and E. Giunchiglia were each partially supported by the projects FAIR (PE00000013) and SERICS (PE00000014), respectively, under the NRRP MUR program funded by the EU - NGEU.

References

Barrett, C., and Tinelli, C. 2018. Satisfiability modulo theories. *Handbook of model checking* 305–343.

Barrett, C.; Fontaine, P.; and Tinelli, C. 2016. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`. Accessed: 2024-01-06.

Billionnet, A. 2003. Using integer programming to solve the train-platforming problem. *Transportation science* 37(2):213–222.

Boleto, G.; Oneto, L.; Cardellini, M.; Maratea, M.; Vallati, M.; Canepa, R.; and Anguita, D. 2021. In-Station Train Movements Prediction: from Shallow to Deep Multi Scale Models. In *29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2021, Online event (Bruges, Belgium), October 6-8, 2021*.

Bylander, T. 1991. Complexity Results for Planning. In *Automated Reasoning*, 6.

Cardellini, M., and Giunchiglia, E. 2025. Temporal Numeric Planning with Patterns. *Proceedings of the AAAI Conference on Artificial Intelligence* 39(25):26481–26489.

Cardellini, M., and Giunchiglia, E. 2026. Symbolic Pattern Temporal Numeric Planning with Intermediate Conditions and Effects. *arXiv* (arXiv:2602.09798).

Cardellini, M.; Maratea, M.; Vallati, M.; Boleto, G.; and Oneto, L. 2021. In-Station Train Dispatching: A PDDL+ Planning Approach. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 450–458.

Cardellini, M.; Giunchiglia, E.; and Maratea, M. 2026. Symbolic pattern planning. *Artif. Intell.* 352:104482.

De Moura, L., and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

Gelfond, M., and Lifschitz, V. 1998. *Action languages*. Linköping University Electronic Press.

Gigante, N.; Micheli, A.; Montanari, A.; and Scala, E. 2022a. Decidability and complexity of action-based temporal planning over dense time. *Artif. Intell.* 307:103686.

Gigante, N.; Micheli, A.; Montanari, A.; and Scala, E. 2022b. Decidability and complexity of action-based temporal planning over dense time. *Artificial Intelligence* 307:103686.

Kautz, H. A., and Selman, B. 1992. Planning as Satisfiability. In Neumann, B., ed., *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings*, 359–363. John Wiley and Sons.

Kloster, O.; Luteberget, B.; Mannino, C.; and Sartor, G. 2025. Displib: a library of train dispatching problems.

Lamorgese, L., and Mannino, C. 2015. An exact decomposition approach for the real-time train dispatching problem. *Operations Research* 63(1):48–64.

Lamorgese, L.; Mannino, C.; Pacciarelli, D.; and Krasemann, J. T. 2018. *Train Dispatching*. Cham: Springer International Publishing. 265–283.

Lindfeldt, O. 2010. *Railway operation analysis: Evaluation of quality, infrastructure and timetable on single and double-track lines with analytical models and simulation*. Ph.D. Dissertation, KTH.

Mannino, C., and Mascis, A. 2009. Optimal real-time traffic control in metro stations. *Operations Research* 57(4):1026–1039.

Marcysiak, A., and Marcysiak, A. 2023. The impact of the COVID-19 epidemic on rail transport in Europe. *Management and Administration Journal* 60(133).

McCarthy, J., and Hayes, P. 1969. Some Philosophical Problems From the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Rodriguez, J. 2007. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological* 41(2):231–245.

Scala, E.; Haslum, P.; Thiebaux, S.; and Ramirez, M. 2016. Interval-Based Relaxation for General Numeric Planning. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, 655–663.

Schutt, A.; Cardellini, M.; Dekker, J. J.; Harabor, D.; Maratea, M.; and Vallati, M. 2025. Constraint-Based In-Station Train Dispatching. In *31st International Conference on Principles and Practice of Constraint Programming (CP 2025)*, 33–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Smith, D. E.; Frank, J.; and Cushing, W. 2008. The ANML language. In *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, volume 31.

Valentini, A.; Micheli, A.; and Cimatti, A. 2020. Temporal Planning with Intermediate Conditions and Effects. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(06):9975–9982.

Zwaneveld, P. J.; Kroon, L. G.; Romeijn, H. E.; Salomon, M.; Dauzere-Peres, S.; Van Hoesel, S. P.; and Ambergen, H. W. 1996. Routing trains through railway stations: Model formulation and algorithms. *Transportation science* 30(3):181–194.