Railway Sciences

Dynamic route optimization in disrupted heterogeneous railway systems via temporal planning

Pollob Chandra Ray, Sabah Binte Noor and Fazlul Hasan Siddiqui
Department of Computer Science and Engineering,
Dhaka University of Engineering & Technology, Gazipur, Bangladesh

Received 6 June 2026
Revised 15 August 2026
Accepted 19 August 2026

Abstract

Purpose – Efficient route optimization is vital for ensuring both safety and punctuality in railway operations. The challenge is particularly pronounced in heterogeneous multi-gauge railway networks, where varying train speeds, stopping patterns and infrastructure compatibility constraints increase coordination complexity. In single-track systems, these challenges are further intensified because trains share the same track and often require frequent track switching. Stochastic disruptions, such as blocked tracks, blocked trains, engine failures and speed slowdowns, add further uncertainty and disrupt the timetable. However, existing studies predominantly focus on high-level timetabling and often omit operational details such as track-switching coordination. Therefore, this study proposes a framework based on temporal planning for route optimization and disruption management in heterogeneous railway systems.

Design/methodology/approach – The proposed framework formulates railway operations as a temporal planning problem, explicitly modeling gauge compatibility constraints and diverse disruption scenarios. It generates conflict-free, timestamped operational plans that provide coordinated schedules together with executable action sequences. To evaluate the framework, we have developed a benchmark of 200 instances with up to 1,000 track points and 120 trains, and evaluated the framework on this benchmark using two state-of-the-art temporal planners and a plan validator.

Findings – The experiments show that the proposed model can generate valid temporal plans across a wide range of heterogeneous railway scenarios with multi-gauge constraints and disruption-recovery requirements.

Originality/value – These findings demonstrate the feasibility of applying temporal planning techniques to disruption-aware railway routing and scheduling problems.

Keywords AI planning, Temporal planning, Railway route optimization, Railway scheduling, Disruption-aware planning, Resource constrained scheduling

Paper type Research article

1. Introduction

Railway is one of the most efficient ways to transport people and goods at scale, providing greater energy efficiency and the ability to carry large volumes of freight at lower cost (Ul Islam *et al.*, 2024). With rising demand, railway infrastructure encounters significant challenges to ensure reliable and timely services due to network congestion (National Academies of Sciences, Engineering, & and Medicine, 2007). Thus, maintaining reliable service requires dynamic route optimization and efficient timetabling to manage network congestion and reduce operational delays.

Railway scheduling determines train timetables with departure and arrival time at each railway track point (Kang, Buhigiro, Sun, & Wu, 2024), while route optimization finds travel paths for trains to minimize travel duration and avoid conflicts (Wang, Song, He, & Song, 2022). Though both problems have decades of research behind them, yet they are hard to solve at scale and known to be NP-hard problem (Leutwiler & Corman, 2023). In addition,

Railway Sciences
Emerald Publishing Limited
e-ISSN: 2755-0915
p-ISSN: 2755-0907
DOI 10.1108/RS-06-2026-0047

real-world systems introduce additional layers of complexity. For instance, heterogeneous railway systems, in which trains have varying speed profiles and stopping patterns, require a higher level of coordination than uniform metro systems (Yan, Bešinović, & Goverde, 2019). Multi-gauge networks amplify this complexity by introducing multiple gauge types for track and train, and require matching gauge types for train movement (Villalba Sanchis, Insa Franco, Martínez Fernández, & Salvador Zuriaga, 2021). While standard gauge (1,435 mm) predominates in Europe and many other regions, several countries operate multi-gauge systems (Grigonis, Kaušylas, & Palevičius, 2025; Villalba Sanchis *et al.*, 2021). For example, Spain railway operates three track gauges: Iberian (1,668 mm) across most of its legacy network, standard (1,435 mm) on high-speed lines and dual-gauge sections that accommodate both Iberian and standard (Villalba Sanchis *et al.*, 2021). Similarly, Bangladesh railway operates three track gauges: meter gauge (1,000 mm), broad gauge (1,676 mm) and dual gauge, which has a combination of both (Bangladesh Railway, 2023). The practical consequence is that gauge type determines which tracks a train can physically use. To drive a train on a specific track, it must follow the gauge compatibility constraints. For instance, a standard gauge train can only run on standard or dual gauge track. Similarly, an Iberian gauge train needs Iberian or dual gauge track. These constraints significantly limit route options, thereby increasing operational complexity.

Collectively, these characteristics define what we refer to as a *heterogeneous* railway system, characterized along four dimensions: (1) *speed heterogeneity*, in which trains with different speed profiles share the same track segments, (2) *stopping-pattern heterogeneity*, in which each express train may follow a distinct stopping pattern and different passenger boarding times at stations, (3) *route-choice flexibility*, in which multiple admissible paths exist between an origin and a destination and (4) *gauge compatibility*, in which track segments and rolling stock are constrained by differing gauges. Most prior studies address only a subset of these dimensions. In contrast, this work addresses all four jointly.

Moreover, railway operations get frequently disrupted by stochastic events such as blocked tracks, blocked trains, engine failures and speed slowdowns (Xiu, Pan, D'Ariano, Zhan, & Peng, 2024). These events cause deviations from the planned railway timetable and require dynamic rescheduling and route optimization to resume normal operations. Numerous studies have been conducted on various aspect of railway operations (discussed in Section 2), such as scheduling, disruption handling and rail station management. However, these studies typically focus on specific aspects of railway operations rather than addressing them in an integrated manner. Most existing approaches also generate optimized schedules without explicitly specifying the executable actions required to realize those schedules. Consequently, critical operational decisions, such as turnout (track-switching) operations, are often left to human operators, increasing the risk of operational errors and safety incidents (Neves, Ribeiro, Grilo, Infante, & Andrade, 2024). For instance, in Bangladesh, human errors accounted for 50.77% of operational incidents during 2022–2023 (Bangladesh Railway, 2023). These limitations raise the need for an integrated approach that can generate feasible and optimized schedules with explicit executable operational plans. To address this need, this study proposes a temporal planning framework for railway route optimization in heterogeneous railway systems.

Automated planning is a fundamental field of artificial intelligence that automatically synthesizes sequences of actions for achieving specified goals from a given initial state. Temporal planning extends this capability to produce timestamped plans by incorporating action durations and temporal constraints, and minimizes the overall operational time (Benton, Coles, & Coles, 2012). These timestamps specify precisely which actions to execute and when, making temporal planning particularly well-suited for railway scheduling and route optimization, where precise timing, coordination and sequencing of train movements are critical. Modern temporal planners can handle the complexity inherent in railway networks efficiently since they employ efficient search strategies (e.g. A*, IDA*) along with domain-independent heuristics. In this work, we model railway operations by encoding safety constraints, gauge compatibility and turnout operations using temporal planning, and

automate the generation of low-level commands necessary to resolve disruptions. We employ PDDL 2.1 (Planning Domain Definition Language) to design our model.

We also systematically design a 200-instance benchmark (discussed in Section 4.3) comprising 100 nominal and 100 disrupted instances distributed across four problem-size categories: small, medium, large and very large. The instances follow a monotonic scaling scheme, with progressively increasing numbers of trains and junctions across the categories, reaching up to 120 trains and 100 junctions in the largest instances. This benchmark enables this study to analyze how the existing planners perform as the network density and disruption severity of the problem increase simultaneously.

We evaluate our proposed framework and benchmark using two state-of-the-art temporal planners: POPF (Coles, Coles, Fox, & Long, 2010) and OPTIC (Benton *et al.*, 2012). Experimental results demonstrate that both planners efficiently generate conflict-free plans, with statistical analysis confirming predictable scaling even in highly disrupted scenarios. Every generated plan is also validated using a temporal validator VAL (Howey, Long, & Fox, 2004), to confirm that the action sequence is consistent and correctly ordered.

The proposed framework has important practical and research implications. Practically, it supports operational dispatching by generating executable plans rather than only revised timetables. These plans specify low-level actions, including train movements, turnout switching, passenger boarding and recovery operations, with explicit start times and durations, while minimizing overall operational time. Disruption recovery is also modeled at the operational level. For example, an engine failure is handled through explicit auxiliary-engine dispatch, attachment and assisted movement. Across the benchmark suite, large-scale disruptions require only 6–13% additional actions, maintaining computational efficiency on commodity hardware. These results establish the practical scope of the framework and motivate the research directions discussed in Section 6.

1.1 An illustrative example

We demonstrate how our framework models railway operations and generates executable plans using a simplified railway network, presented in Figure 1. The network consists of six track points (*A, B, C, D, E, F*), where *B* and *E* are junction (turnout) points. Junction *B* can connect either to point *C* or *D* (currently connected to *D*), while junction *E* can connect to *C* or *D* (currently connected to *D*).

The network also includes a heterogeneous *multi-gauge* infrastructure. Track segments *A*-*B* and *E*-*F* are dual gauge, allowing both meter-gauge and broad-gauge trains. Segments *B*-*D* and *D*-*E* support only broad-gauge trains, while segments *B*-*C* and *C*-*E* support only meter-gauge trains. Station *S1*, located between junctions *B* and *E*, consists of two parallel platform points *C* and *D* where trains stop for passenger boarding. The network has two trains, train *t1* (broad gauge) and train *t2* (meter gauge).

The task is to move train *t1* from point *A* to *F*, and to move train *t2* from point *F* to *A*. Train *t1* also needs to stop at point *D* for a 2-min boarding operation, while train *t2* requires a 3-min boarding at point *C*. Assume trains *t1* and *t2* travel at 1 km/min (60 km/h) and 1.1 km/min (66 km/h), respectively. Turnout operations require 1 minute to switch track alignment. Table 1 presents a temporal plan, consisting of timestamped train movements, turnout operations and passenger boarding actions, each specified by its start time and duration (mm).

In the temporal plan in Table 1, $DriveTrain(t1, A, B, broad)$ starts at 00:00 with a duration of 60:00, indicating that train *t1* departs point *A* at time 0 and arrives at point *B* after 60 minutes. The parameter *broad* specifies that *t1* traverses the segment on a compatible broad-gauge track. The plan also illustrates the temporal coordination the model provides. Both trains and the turnout at *E* act concurrently from time zero (00:00), and the junctions are reconfigured as the plan unfolds: the turnout at *E* is first switched toward *C* so that *t2* can

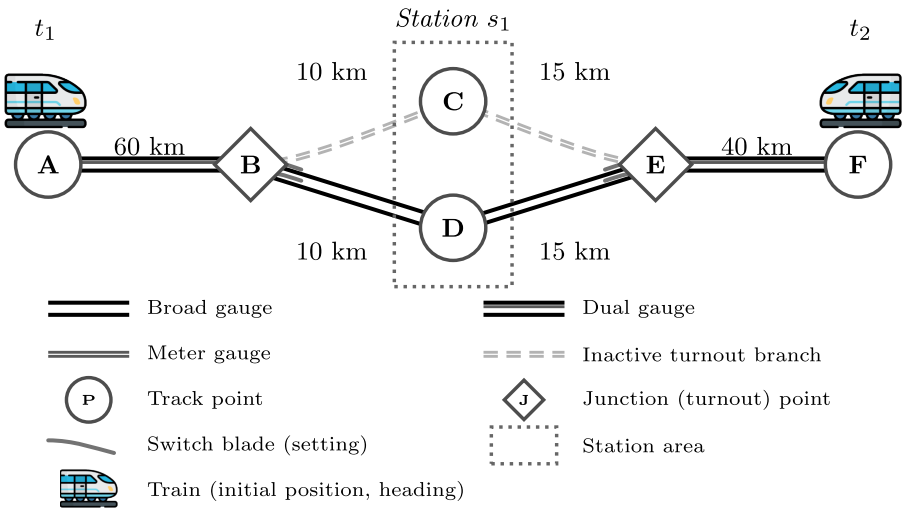

Figure 1. Railway network with six track points (A–F). Circles denote track points, diamonds denote the junction (turnout) points B and E, with the curved switch blades showing the current turnout setting toward D, and dashed lines indicating the inactive turnout branches. Line styles distinguish the gauge of each segment: heavy line pairs denote broad-gauge tracks, lighter line pairs denote meter-gauge tracks and combined line segments denote dual-gauge tracks (rendered here with three lines for illustration). The dotted rectangle marks station s_1, comprising the platform points C and D. Track distances are labeled along each segment. Train t_1 (broad gauge, 60 km/h) departs from A and train t_2 (meter gauge, 66 km/h) from F, with icons indicating initial positions and headings

Table 1. A temporal plan for the railway routing problem of Figure 1

Start (mm:ss)	Action	Duration (mm:ss)
00:00	DriveTrain(*t1*, *A*, *B*, *broad*)	60:00
00:00	DriveTrain(*t2*, *F*, *E*, *meter*)	36:22
00:00	Turnout(*E*, *D*, *C*)	01:00
36:22	DriveTrain(*t2*, *E*, *C*, *meter*)	13:38
50:00	Turnout(*E*, *C*, *D*)	01:00
50:00	BoardPassengers(*t2*, *s1*, *C*)	03:00
60:00	DriveTrain(*t1*, *B*, *D*, *broad*)	10:00
70:00	Turnout(*B*, *D*, *C*)	01:00
70:00	BoardPassengers(*t1*, *s1*, *D*)	02:00
71:00	DriveTrain(*t2*, *C*, *B*, *meter*)	09:05
72:00	DriveTrain(*t1*, *D*, *E*, *broad*)	15:00
80:06	DriveTrain(*t2*, *B*, *A*, *meter*)	54:33
87:00	DriveTrain(*t1*, *E*, *F*, *broad*)	40:00

Note(s): Start times and durations are given in minutes and seconds (mm). The plan has a makespan of 134 minutes and 39 seconds

reach its boarding point, and switched back toward *D* at 50:00 so that *t1* can later pass through *E* on its route to *F*, each switch occupying the junction for its one-minute duration. This simple example demonstrates how the generated plan provides an executable operational schedule with precise action timing and satisfying gauge compatibility, station operations and turnout constraints.

1.2 Contributions

This article makes the following contributions, organized into modeling, benchmarking and empirical evaluation:

(1) *A temporal planning model for heterogeneous railway routing:* We formalize dynamic route optimization in multi-gauge networks as a temporal planning problem that jointly captures (1) gauge compatibility routing constraint, (2) turnouts as durative, resource-consuming actions, with switching time and junction unavailability factored into the temporal optimization and (3) track segments and points as exclusive temporal resources, yielding collision-free schedules by construction.

(2) *Integrated disruption management:* We extend the temporal planning framework to support four classes of railway disruptions: blocked trains, blocked tracks, engine failures and slowdowns, via dedicated state predicates and durative recovery actions, including an auxiliary-engine rescue pipeline (dispatch, attachment, assisted movement).

(3) *A scalable benchmark for heterogeneous railway planning:* We construct a 200-instance benchmark comprising 100 nominal and 100 disrupted instances across four progressively larger network categories, scaling up to 1,000 track points and 120 trains with increasing disruption severity.

(4) *A complete, open PDDL implementation and empirical evaluation:* We provide the complete PDDL 2.1 domain, 200 benchmark instances and instance generator to reproduce the experimental results. We evaluate the benchmark using the OPTIC and POPF temporal planners and validate all generated plans using VAL. The evaluation compares planner coverage and runtime across problem sizes, analyzes delay according to disruption type and quantifies disruption resilience relative to nominal operations.

This work extends an earlier conference paper by introducing additional disruption scenarios, a large-scale benchmark problem set and an expanded experimental evaluation.

1.3 Organization

The remainder of this article is organized as follows. In Section 2, we discuss the existing approaches to railway scheduling and optimization in the literature. Section 3 provides the necessary background and semantics on temporal planning, and its modeling using PDDL. In Section 4, we present the proposed framework, including the design of the temporal planning domain and the formulation of railway planning problems. Following that, we discuss experimental evaluation and analysis in Section 5 and conclude with our remarks in Section 6.

2. Related works

Researchers have proposed a wide range of optimization models and algorithms for railway scheduling, aiming to minimize delays, reduce waiting times, lower operational costs, and enhance system efficiency and service reliability. These approaches can be broadly categorized into Maximum Satisfiability (MaxSAT), Mixed Integer Linear Programming (MILP), metaheuristic algorithms, and automated planning. MILP and MaxSAT formulate railway scheduling as mathematical optimization problems that produce optimal or near-optimal conflict-free timetables, whereas metaheuristic methods search efficiently for high-quality scheduling solutions under complex constraints. In contrast, temporal planning directly generates executable, timestamped action sequences that naturally capture action durations and concurrency. The remainder of this section reviews representative studies from each of these four methodological categories and discusses how they address the heterogeneity dimensions introduced in Section 1.

Maximum Satisfiability (MaxSAT) formulations have recently gained considerable attention for optimizing scheduling under complex operational constraints. Lemos, Gouveia, Monteiro, and Lynce (2024) proposed an iterative MaxSAT-based framework for the Swiss Federal Railways (SBB) that minimizes delays and routing costs while handling track blockages and speed reductions through rerouting, waiting strategies and velocity adjustments. Gouveia, Albino, and Saldanha (2025) developed a MaxSAT formulation to generate conflict-free timetables while preserving limited flexibility in route choice, and tested their model on real data from SBB and the Washington Metro. Although these approaches effectively generate conflict-free schedules, with Lemos *et al.* (2024) additionally supporting rerouting and speed adjustment, they generally assume relatively uniform train characteristics and do not jointly model stopping-pattern variation and gauge compatibility across heterogeneous rolling stock. Moreover, their solutions remain primarily timetable-based rather than action-level operational plans.

MILP has also been a widely adopted method for railway scheduling and rescheduling. Chai *et al.* (2024) built a branch-and-cut framework for urban rail that jointly optimizes train sequencing, station timing and coupling operations under time-varying passenger demand. Zhuo, Miao, Meng, Yang, and Shang (2024) took a similar demand-driven angle, adapting both schedules and train compositions as passenger loads shift. Ji *et al.* (2024) developed a two-stage stochastic optimization model to address uncertainty in railway maintenance durations. Additional MILP and ILP-based methods have been proposed for disruption management scenarios, including track blockages, rolling stock failures and timetable adjustments (Acuna-Agost, Michelon, Feillet, & Gueye, 2011; Fischetti & Monaci, 2017; Li, Zhao, Peng, Wang, & Zhong, 2025; Veelenturf, Kidd, Cacchiani, Kroon, & Toth, 2016; Xiu *et al.*, 2024). These models provide powerful optimization capabilities and, in several cases, achieve global optimality for the problems they target. However, their formulations generally focus on timetable and sequencing decisions over a predefined infrastructure representation. Heterogeneity dimensions such as multi-gauge compatibility and route-level diversity typically fall outside these formulations. Consequently, the network-wide, action-level planning problem under combined speed, stopping-pattern, route and gauge heterogeneity remains largely unaddressed.

Metaheuristic approaches have also shown strong performance on scheduling and rescheduling instances (Fang, Yang, & Yao, 2015; Narayanaswami & Rangaraj, 2011). Genetic Algorithms (Nitisiri, Gen, & Ohwada, 2019), Simulated Annealing (Zhang & Ni, 2022), Large Neighborhood Search (Zhang & Ni, 2022) and Ant Colony Optimization (Coviello, Medeossi, Nygreen, Pellegrini, & Rodriguez, 2023) have been employed to navigate the nonlinear solution spaces of large-scale networks, with reported successes in reducing passenger waiting times and operating cycles under fluctuating demand. Representative examples include the Problem Space Search metaheuristic of Albrecht, Panton, and Lee (2013), which reschedules timetables under scheduled maintenance disruptions while preserving schedule quality, and the hybrid framework of Dündar and Şahin (2013), which couples a genetic algorithm for conflict resolution with an artificial neural network emulating dispatcher decision. In principle, such algorithms are flexible enough to encode heterogeneous infrastructure and rolling-stock constraints within their objective and feasibility functions. However, these studies frequently rely on simplified network abstractions (i.e. uniform gauge compatibility and homogeneous speed profiles) in order to keep the search space tractable. This reflects the particular modeling choices made in those studies, not any fundamental limitation of metaheuristics as a class. Consequently, while the paradigm itself remains a viable candidate, existing metaheuristic formulations in the literature do not directly produce feasible, action-level plans for the combined heterogeneity addressed here.

Automated planning offers an alternative paradigm that generates explicit action sequences for solving a particular task, and has been applied across several domains relevant to ours. Nyporko and Chrpa (2025) modeled industrial production and

manufacturing with PDDL 2.1, selecting and scheduling activities on the resources that perform them, with each action representing an elementary production step. Yang and Liang (2022) applied automated planning to emergency-material logistics, regulating the transport and dispatch of supplies to demand points so as to reduce the losses caused by shortages. A temporal planning formulation of the temporally constrained journey problem was proposed by Caff, Di Mauro, and Scala (2014), producing solutions that satisfy cost, trajectory and maximum-travel-time constraints. Within the railway domain specifically, Cardellini, Maratea, Vallati, Boleto, and Oneto (2021) developed a numeric planning-based framework for train dispatching trains inside stations. Louadah *et al.* (2021) used temporal planning to schedule maintenance in depots by translating ontological knowledge and SWRL rules into PDDL. These works demonstrate that planning can coordinate concurrent activities and optimize localized operations. However, they are confined to station or depot level settings and do not extend to network-wide route planning, heterogeneous train constraints or integrated disruption handling, precisely the scope our framework targets.

Across these advancements, two critical gaps remain. First, most studies incorporate only a subset of the heterogeneity dimensions defined in Section 1, while multi-gauge compatibility and joint route choice are rarely modeled together. Second, the produced output is a conflict-free timetable, and disruption is handled by reactively adjusting that timetable, rather than by proactively generating detailed, timestamped execution plans of low-level actions. This study addresses both gaps by introducing a temporal planning framework that explicitly accounts for heterogeneity of railway and that produces precise, actionable recovery plans for complex disruptions.

3. Preliminaries

This section presents the theoretical foundations of temporal planning that underpin our approach. It introduces the key concepts and definitions of temporal planning and the PDDL required for modeling the proposed framework.

3.1 Temporal planning

Automated planning and scheduling is a core subfield of artificial intelligence concerned with the automatic generation of action sequences that transform a given initial state into one satisfying specified goal conditions (Ghallab, Nau, & Traverso, 2004). Learning-based approaches derive decision-making behavior from data through statistical training, whereas automated planning relies on explicit, symbolic models of actions, states and goals, and computes solutions through search and reasoning over these models. A planning system (planner), such as Optic or POPF, takes as input a formal description of the initial state, the available actions and the goal conditions, and produces a plan (a structured sequence of actions) that achieves the goals when executed.

In classical planning, actions are assumed to be instantaneous and sequentially executed. The environment is typically modeled as fully observable, deterministic, static (i.e. state changes occur only through agent actions) and finite. However, modeling many real-world domains that involve action durations, temporal dependencies or concurrent activities can be challenging within the classical planning framework. Railway operations, for example, involve activities such as train movements and turnout operations that require explicit representation of time and resource constraints.

Temporal planning extends classical planning by explicitly incorporating time into the model (Cenamor, Vallati, & Chrpa, 2019). Intuitively, a temporal planning problem (Definition 1) specifies the system state using logical and numeric variables, a set of durative actions that modify the state, an initial configuration of the system (given by an assignment over both fluent types) and a set of goal conditions that the planner must

achieve. A durative action (Definition 2) represents an activity that unfolds over a time interval bounded by a start and an end instant. Three temporally distinguished sets of conditions govern its applicability: conditions required at the start instant, conditions required to hold throughout the entire interval and conditions required at the end instant. Correspondingly, its effects are partitioned into those applied at the start instant and those applied at the end instant, reflecting how the action's impact on the state may unfold at different points in its execution.

Definition 1. A *temporal planning problem* is defined as a 5-tuple $P_T = \langle F, X, A_d, I, G \rangle$, where:

(1) F is a finite set of propositional fluents,

(2) X is a set of real-valued numeric fluents,

(3) A_d is a finite set of *durative actions*,

(4) $I \subseteq F \cup X$ represents the initial state, given by a truth assignment to the fluents in F and a real-valued assignment to the numeric fluents in X and

(5) $G \subseteq F \cup X$ represents the goal condition, expressed as a set of propositional fluents that must hold (optionally together with numeric constraints over X)

Definition 2. A *durative action* $a \in A_d$ is defined as $a = \langle pre_{\vdash}(a), pre_{\leftrightarrow}(a), pre_{\dashv}(a), eff_{\vdash}(a), eff_{\dashv}(a), dur(a) \rangle$, where

(1) $pre_{\vdash}(a)$ are *start conditions*, required to hold at the instant the action starts,

(2) $pre_{\leftrightarrow}(a)$ are *over-all conditions*, required to hold continuously throughout the action's execution duration,

(3) $pre_{\dashv}(a)$ are *end conditions*, required to hold at the instant the action ends,

(4) $eff_{\vdash}(a)$ are *start effects (changes)*, applied at the instant the action starts,

(5) $eff_{\dashv}(a)$ are *end effects (changes)*, applied at the instant the action ends and

(6) $dur(a) \in R^{+}$ denotes the action duration.

Unlike classical planning, a temporal plan is not just a sequence of actions but a time-stamped schedule of (t_i, a_i) pairs, where t_i denotes the start time of action a_i (Haslum, Lipovetzky, Magazzeni, & Muise, 2019). A temporal plan is valid if and only if (1) all preconditions are satisfied at their required times, (2) over all (invariant) conditions hold throughout $[t_i, t_i + dur(a_i)]$, no conflicting effects occur simultaneously and the goal G is satisfied at plan completion, $t_{end} = max_i(t_i + dur(a_i))$. The makespan (Definition 3) of a temporal plan measures the total time required to complete all actions, from the start of the earliest action to the completion of the latest one.

Definition 3. The *makespan* of a temporal plan Π_T is the total execution time of the plan, defined as:

$$makespan(\Pi_T) = \max_{(a_i, t_i) \in \Pi_T} (t_i + dur(a_i))$$

To formalize a planning problem and generate valid temporal plans, we need to describe the planning environment, such as states, actions and goals. PDDL (Haslum *et al.*, 2019) is the most widely adopted language for this purpose.

3.2 Planning Domain Definition Language

An Automated planning system requires a formal representation of the *initial state*, the *actions* to enable transitions between states and the *goal conditions* to generate a valid plan. For the past two decades, the planning community has widely adopted PDDL as the standard for modeling planning problems. PDDL divides a planning problem into two components: a domain and a problem instance. The domain captures the general structure of the planning environment, including state predicates and action definitions, while the problem instance specifies a particular scenario by defining the initial state and the desired goal conditions.

PDDL 2.1 extends classical PDDL by introducing temporal and numeric features (Fox & Long, 2003). In particular, it supports durative actions with explicit durations, temporal qualifiers (*at start*, *over all*, *at end*), numeric fluents and arithmetic expressions, and optimization metrics (e.g. makespan minimization).

We describe the syntax of PDDL 2.1 using the *domain* definition and the *problem instance* of a toy railway system, presented in Listings 1 and 2, respectively.

3.2.1 Domain definition. A temporal planning domain specifies the object types, state predicates, numeric fluents and durative-action schemas of the environment. Each durative action declares typed parameters, a duration, temporally annotated conditions that must hold *at start*, *over all* (as invariants) or *at end* of execution, and effects applied at the start or end of the action. Listing 1 presents a simplified railway domain illustrating these concepts. The domain defines two object types, *train* and *location*, two predicates, at and connected, representing the train's current location and the connectivity between locations, respectively, a numeric fluent total-cost and a single durative action, *move*.

Listing 1. Temporal Domain definition for a toy railway system.

```
(define (domain simple_temporal_railway)
  (:requirements :strips :typing :durative-actions :fluents)
  (:types train location)
  (:predicates
    (at ?t - train ?l - location)
    (connected ?from - location ?to - location))
  (:functions (total-cost))
  (:durative-action move
    :parameters (?t - train ?from - location ?to - location)
    :duration (= ?duration 5)
    :condition (and
      (at start (at ?t ?from))
      (at start (connected ?from ?to)))
    :effect (and
      (at start (not (at ?t ?from)))
      (at end (at ?t ?to))
      (at end (increase (total-cost) 10)))
  )
)
```

The action has a fixed duration of five-time units and requires the train to be at the origin location when execution begins. In addition, the origin and destination locations need to be connected. At the start of execution, the action removes the train from the origin and places it at the destination upon completion. It then increases the accumulated travel cost by 10 units at the end of the action. Although this example uses a fixed action duration for simplicity, PDDL 2.1 also permits duration expressions over numeric fluents. Our proposed railway domain (Section 4) computes action durations dynamically from track distances, train speeds and slowdown factors.

3.2.2 Problem definition. A problem instance declares the objects, the initial state and the goal of a concrete scenario, providing the data from which the planner instantiates the domain actions. Listing 2 shows a toy instance for the domain in Listing 1. It defines one train, three locations, the initial location of the train, the connectivity between adjacent locations and initializes the accumulated cost to zero. The goal specifies that the train must reach location *C*, while the metric directs the planner to minimize the total action cost. Alternatively, specifying the special fluent *total-time* as the optimization metric minimizes the plan makespan. Temporal planners then search for a temporally valid plan optimizing the given objective.

Listing 2. Problem definition for the railway domain in Listing 1.

```
(define (problem toy_instance)
  (:domain simple_temporal_railway)
  (:objects
    train1 - train
    A B C - location)
  (:init
    (at train1 A)
    (connected A B)
    (connected B C)
    (= (total-cost) 0))
  (:goal (at train1 C))
  (:metric minimize (total-cost))
)
```

A temporal plan for this instance consists of two sequential durative actions. Since locations *A* and *C* are not directly connected, the train must travel via the intermediate location *B*. Each plan entry specifies the action's start time, its ground instantiation and its duration. The temporal planners used in this work produce plans in the following format:

$$\pi = \{(0.0, Move(train1, A, B), 5.0), (5.0, Move(train1, B, C), 5.0)\}.$$

This plan has a makespan of 10 time units and a total cost of 20, since each action has a duration of 5 time units and incurs a cost of 10 units.

Such expressive capabilities of PDDL 2.1, including durative actions with temporal annotations and numeric fluents, provide a solid basis for the railway domain model developed in this study.

4. Methodological framework and problem formulation

This section presents the proposed *Disruption Aware Railway Temporal Planning (DART)* framework, which models railway systems as a temporal planning problem. The framework generates detailed, time-stamped plans for coordinating train movements, infrastructure operations and disruption recovery while satisfying gauge compatibility and operational constraints.

To address operational challenges in heterogeneous railway networks, DART incorporates a disruption taxonomy and corresponding recovery strategies for each disruption type. The railway network, rolling stock, infrastructure components and disruption scenarios are encoded in PDDL 2.1, enabling temporal planners to generate executable plans that dynamically adapt to operational disruptions.

The remainder of this section presents the disruption taxonomy, the formal temporal planning model, the railway action models and the benchmark generation methodology.

4.1 Disruption taxonomy and recovery strategies

Railway operations in heterogeneous multi-gauge networks are frequently affected by stochastic disruptions that deviate from nominal schedules, thus requiring recovery planning. DART formalizes four primary disruption types, each represented by dedicated predicates and durative recovery actions (Section 4.2). These disruptions also form the basis for our benchmark instances in Section 4.3:

(1) *Blocked Track Disruption* occurs when a railway track segment becomes unavailable for train movements due to accidents, infrastructure failures or maintenance activities. Recovery strategies in our framework include (1) waiting at the current position until the track is cleared or (2) rerouting via gauge-compatible alternative paths using turnout operations.

(2) *Blocked Train Disruption* occurs when a train is temporarily immobilized (e.g. due to logistical issues, accidents or crew unavailability). When this disruption occurs, the affected train must remain at the disruption point until the blockage is resolved, while other trains continue operations

(3) *Slowdown Disruption* reduces train speeds over specific track segments due to external conditions such as adverse weather. Travel time on a disrupted segment is computed as:

$$\tau\left(ni, n_j\right) = d\left(ni, n_j\right) / \left[\left(1 - \alpha\left(ni, n_j\right)\right) \cdot v(t)\right] \tag{1}$$

where τ denotes the train (or engine) traversing the disrupted segment (ni, n_j), $v(t)$ its speed and $\alpha(ni, n_j) \in [0, 1)$ the slowdown factor applied to that segment (e.g. $\alpha = 0.2$ for a 20% speed reduction). Recovery strategies include traversing with increased travel time or rerouting to minimize makespan.

(4) *Engine Failure Disruptions* occur when a locomotive becomes inoperative. Recovery involves dispatching an auxiliary engine to the affected train and coupling it to resume operations.

Together, these four disruption categories constitute a comprehensive operational stress model for the heterogeneous railway networks.

4.2 Temporal railway task formulation

We formulate railway route optimization and disruption management as a temporal planning problem. The formulation assumes that the railway system is fully observable and deterministic, disruption durations (e.g. blocked tracks, blocked trains and slowdowns) are

known a priori, and trains travel at constant speeds without acceleration or deceleration. For clarity and accessibility, we present the railway planning model using a first-order logical formalization, allowing readers to understand the underlying model independently of its PDDL implementation. The corresponding PDDL domain and problem specifications are then derived from this formalization.

Definition 4. A railway temporal planning task is represented as $P = \langle F_R, X_R, A_R, I_R, G_R \rangle$, where:

(1) F_R is the finite set of grounded propositional facts over the predicate set P_R, describing the railway states.

$$P_R = \{train\_at, engine\_at, connected, free, track\_accessible, junction\_point, turnout\_alternatives, train\_gauge, track\_gauge, engine\_gauge, track\_blocked, track\_clear, engine\_damaged, engine\_functional, engine\_free, engine\_attached, platform\_at, boarding\_point, passenger\_boarded, up.down, heading\_up, heading\_down\}$$

(2) X_R is the finite set of grounded numeric fluents over the function set X_R, representing quantitative properties of the railway system.

$$X_R = \{distance, train\_speed, engine\_speed, boarding\_time, turnout\_time, slowdown, train\_blockage\_time, track\_clear\_time, engine\_attach\_time\}$$

(3) A_R is the finite set of durative actions representing railway operations.

$$A_R = \{DriveTrain, DriveEngine, DriveAssistedTrain, BoardPassengers, Turnout, ResolveTrainBlockage, ClearBlockedTrack, AttachEngine, DriveEngineToDamagedUpTrain, DriveEngineToDamagedDownTrain\}$$

(4) I_R denotes the initial railway state and

(5) G_R specifies the goal conditions that every valid plan must satisfy.

The goal of the planning task is to ensure that every train reaches its designated destination while completing all required passenger-service operations. Depending on the disruption scenario, additional recovery conditions, such as clearing blocked tracks or recovering disabled trains, may also be required. Accordingly, the goal condition is defined as:

$$G_R = (\wedge_{t \in T} train\_at(t, Dest(t))) \wedge (\wedge_{t \in T} passengers\_boarded(t))$$

Among all valid temporal plans satisfying the goal conditions, the planner seeks the schedule with the minimum makespan,

$$min\, Makespan(\pi),$$

where π denotes the generated temporal plan.

The following subsections instantiate this formulation in detail, from the underlying type system and state predicates through the durative action schemas and temporal resource constraints, to the construction of a concrete problem instance.

4.2.1 Modeling decisions. This section summarizes the key modeling decisions that shape the predicates and action schemas introduced below, balancing expressiveness against planning tractability.

(1) *Segments and points as exclusive temporal resources:* Rather than checking collisions post hoc, track segments and points are made temporarily unavailable for the duration a vehicle occupies them, so every plan is collision-free by construction rather than by validation.

(2) *Declarative gauge compatibility.* Gauge type is attached to trains, engines and track segments as static facts, and every movement action's preconditions require a matching gauge. This keeps gauge reasoning local to each action rather than requiring a separate consistency check over the whole route.

(3) *Disruptions as explicit states, not stochastic events.* Blocked tracks, blocked trains, engine damage and slowdowns are represented as ordinary state predicates and a fluent, not sampled during execution. This lets a temporal planner reason about recovery the same way it reasons about nominal operations, using the same search machinery.

(4) *Directional dispatch for engine rescue.* Auxiliary-engine dispatch is split into an up-bound and a down-bound schema rather than one direction-agnostic action, so that a rescue engine is only ever routed to approach a disabled train against its direction of travel, matching how rescue operations work in practice.

Together, these decisions keep the domain compact while preserving the correctness guarantees the following action schemas rely on.

4.2.2 World representation. This subsection defines the type hierarchy and state predicates used to model the railway network, the operational entities within it and the numeric fluents governing action durations.

4.2.2.1 Type system. The domain is organized over five object types: *track-point, train, gauge-type, engine* and *station.* Track points form the nodes of the railway network and, together with the segments connecting them, define its topology. Trains and auxiliary engines represent the rolling stock; each train and engine has an associated gauge type and a cruising speed. Stations group the platform points where passenger boarding occurs. Gauge types capture the physical gauge classifications (meter, broad, dual) that constrain which trains may use which segments.

4.2.2.2 State predicates. The predicates P_R collectively define the operational world state of the railway environment. They are organized into five groups by the aspect of the railway operations they describe.

(1) *Spatial configuration.* Predicates train_at and *engine_at* indicates the location of trains and auxiliary engines at track points; platform_at and *boarding_point* predicates identify where a train must stop for service, and *passengers_boarded* records that service is complete; *junction_point* marks a point as a turnout.

(2) *Infrastructure connectivity.* The Predicate *free* marks a point as unoccupied; *connected* reflects the currently active turnout routing between two points; *track_accessible* marks a segment as available for traversal, and is withdrawn while a vehicle occupies it, making segments exclusive temporal resources; *turnout_alternatives* records the two routing choices available at a junction.

(3) *Gauge compatibility.* Predicates *train_gauge*, *engine_gauge* and *track_gauge* attach a gauge type to a train, an auxiliary engine and a track segment, respectively, so that movement actions can enforce compatibility directly in their preconditions.

(4) *Disruption state.* Predicates *track_blocked* and *track_clear* are complementary facts describing whether a segment is passable; *train_blocked* marks an immobilized train; *engine_damaged*, *engine_free* and *engine_attached* track a locomotive failure and its recovery via an auxiliary engine.

(5) *Directional coordination.* The *up* and *down* predicates orient a segment; *heading_up* and *heading_down* record a train's or engine's current travel direction, so a rescue engine can be dispatched against the disabled train's heading.

(6) *Constants.* A set of numeric fluents X_R governs action durations and remains fixed for a given instance. For instance, the *distance* and *train_speed*, *engine_speed* fluents determine travel time; *boarding_time* and *turnout_time* fix service and switching durations; *slowdown* scales travel time under a weather- or track-condition disruption; and *train_blockage_time*, *track_clear_time* and *engine_attach_time* fix the duration of the corresponding recovery actions. Each is computed from infrastructure and vehicle properties rather than hardcoded, so a single domain covers heterogeneous networks without per-instance modification.

This vocabulary (types, predicates and fluents) forms the basis for the durative action schemas introduced in the following subsections.

4.2.3 Durative actions. The proposed railway domain is modeled using ten durative actions to represent operational activities and disruption recovery procedures. We categorize the railway operations into four types: *movement operations*, *passenger service operations*, *disruption recovery operations* and i*nfrastructure operations*.

4.2.3.1 Movement operations. The *DriveTrain* and *DriveEngine* actions (Table 2) model the traversal of trains and auxiliary engines between railway points. We describe *DriveTrain* in detail as the representative example and highlight only the distinguishing features of

Table 2. Movement-operation action schemas

Action	Preconditions *ann.*	*predicate*	Effects *ann.*	*predicate*	Duration
DriveTrain (t, n_i, n_j, g)	*Start*	*train_at*(t, n_i) *train_gauge*(t, g) *track_gauge*(n_i, n_j, g) $\neg$ *train_blocked*(t, n_i) $\neg$ *engine_damaged* (t,n_i) *track_clear*(n_i, n_j) *track_accessible*(n_i, n_j)	*Start*	$\neg$ *train_at*(t, n_i) *free*(n_i) $\neg$ *track_accessible*$(n_i, n_j$ $\neg$ *track_accessible*(n_j, n_i)	$\tau_{v(t)}(n_i, n_j)$
	Over-all	*connected*(n_i, n_j) *free*(n_j)	*End*	*train_at*(t, n_i) $\neg$ *free*(n_j)	
	End	*free*(n_j)		*track_accessible*(n_i, n_j) *track_accessible*(n_j, n_i)	
DriveEngine (e, n_i, n_j, g)	*Start*	*engine_at*(e, n_i) *engine_free*(e) *engine_gauge*(e, g) *track_gauge*(n_i, n_j, g) *track_clear*(n_i, n_j) *track_accessible*(n_i, n_j)	*Start*	$\neg$ *engine_at*(e, n_i) *free*(n_i) $\neg$ *track_accessible*(n_i, n_j) $\neg$ *track_accessible*(n_j, n_i)	$\tau_{v_a(e)}(n_i, n_j)$
	Over-all	*connected*(n_i, n_j) *free*(n_j)	*End*	*engine_at*(e, n_i) $\neg$ *free*(n_j)	
	End	*free*(n_j)		*track_accessible*(n_i, n_j) *track_accessible*(n_j, n_i)	

Note(s): Parameter shorthands: 't' a train, 'e' an auxiliary engine; n_i, n_j track points, 'g' a gauge type. $\tau_u(n_i, n_j)$ denotes the traversal time of segment (n_i, n_j) at speed 'u' (Eq. 1)

subsequent actions. The parameters of the action *DriveTrain* are a train *t*, a track segment defined by the points n_i and n_j, and a gauge type *g*. The temporal conditions of *DriveTrain* ensure that, at initiation, the train and track are gauge-compatible, the segment is clear and accessible, and the train is neither blocked nor operating with a damaged engine. Since OPTIC and POPF do not support negative preconditions, the ¬ *train_blocked(t, n_i)*, ¬ *engine_damaged(t, n_i)* conditions are encoded using positive predicates *train_clear(t, n_i)* and *engine_functional(t, n_i)*. The destination point must remain free throughout execution and upon completion, preventing conflicting occupancy. The temporal effects mark the segment inaccessible in both directions at the start of the action, avoiding simultaneous use. Upon completion, the destination is marked occupied and segment accessibility is restored in both directions.

4.2.3.2 Passenger service operations. Passenger service activities are represented through the *board-passengers* action (Table 3), which ensures that departure occurs only after boarding is complete. Boarding time is defined by $\beta(t, s)$, the time (in minutes) required for train *t* to board passengers at station *s*.

4.2.3.3 Disruption recovery operations. To support disruption recovery, *drive-engine-to-damaged-up-train and drive-engine-to-damaged-down-train* actions dispatch auxiliary engines toward immobilized trains while respecting directional and connectivity constraints. These actions are similar in structure to movement operations, presented in Table 4. The *attach-engine* action (Table 4) models the coupling of an auxiliary engine to a damaged train on a track segment. Both entities (the corresponding engine and train) must be co-located throughout the operation, enforced via over all conditions. The engine is marked unavailable at the start *(not (engine-free ?en))*, and upon completion *engine-attached* is asserted, enabling the subsequent *drive-assisted-train* action. Once attached, the *drive-assisted-train* action, shown in Table 4, enables the damaged train to move, with the auxiliary engine's speed determining the duration. The original train engine remains nonfunctional, which is reflected at the destination.

Operational disruptions are resolved using two dedicated recovery actions. The *resolve-train-blockage* action restores mobility of blocked trains, while *clear-blocked-track* models infrastructure restoration following track obstructions.

4.2.3.4 Infrastructure operations. Infrastructure management primarily involves the *turnout* action (Table 5), which switches routing at junction points. The active connection is severed at initiation, while the alternative is established at completion, modeling physical switching delays via *turnout-time*.

Together, these durative actions provide an integrated temporal representation of railway movement, service operations, infrastructure management and disruption recovery, enabling the planner to generate executable and disruption-resilient railway schedules.

4.2.4 Problem instance modeling. Each problem instance instantiates $\langle I_R, G_R \rangle$ for a concrete scenario over P_R. The initial state $I_R \subseteq F_R$, together with the fluents X_R, fixes train positions and gauges, the network topology and segment gauges, junction alternatives, platform and boarding assignments, distances, speeds and service times; the goal follows the

Table 3. Passenger-service action schema

Action	Preconditions *ann.*	*predicate*	Effects *ann.*	*predicate*	Duration
BoardPassenger *(t, s, n)*	*Start*	*train_at(t, n)* *boarding_point(t, n)* *platform_at(n, s)*	*End*	*passengers_boarded(t,n)*	$\beta(t, n)$
	Over-all	*train_at(t, n)*			

Note(s): *s* denotes a station; other shorthands as in Table 2

Table 4. Disruption-recovery action schemas

Action	Preconditions *ann.*	*predicate*	Effects *ann.*	*predicate*	Duration
DriveEngine ToDamaged UpTrain *(e, t, n_i, n_j,g)*	*Start*	*engine_at(e, n_i) up(n_i, n_j)* *engine_gauge(e, g)* *train_at(t, n_j)* *engine_free(e)* *track_gauge(n_i, n_j, g)* *train_gauge(t,g)* *engine_damaged(t, n_j)* *heading_up(t)* *heading_down(e)* *track_clear(n_i, n_j)* *track_accessible(n_i, n_j)*	*Start*	¬ *engine_at(e, n_i) free(n_i)* ¬ *track_accessible(n_i, n_j)* ¬ *track_accessible(n_j, n_i)*	$\tau_{v_a(e)}(n_i, n_j)$
	Over-all	*connected(n_i, n_j)* *train_at(t, n_j)*	*End*	*engine_at(t, n_j)* *heading_up(e)* ¬ *heading_down(e)* *track_accessible(n_i, n_j* *track_accessible(n_j, n_i)*	
AttachEngine *(e, t, n)*	*Start*	*engine_at(e, n)* *train_at(t, n)* *engine_free(e)* *engine_damaged(t, n)*	*Start*	¬ *engine_free(e)*	$\mu(e)$
	Over-all	*engine_at(e, n)* *train_at(t, n)*	*End*	*engine_attached(e, t)*	
DriveAssisted Train *(e, t, n_i, n_j, g)*	*Start*	*engine_attached(e, t)* *train_at(t, n_i)* *engine_at(e, n_i)* *engine_gauge(e, g)* *track_gauge(n_i, n_j, g)* *track_clear(n_i, n_j)* *track_accessible(n_i, n_j)* ¬ *train_blocked(t, n_i)*	*Start*	¬ *train_at(t, n_i) free(n_i)* ¬ *engine_at(e, n_i)* ¬ *engine_damaged(t, n_i)* ¬ *track_accessible(n_i, n_j)* ¬ *track_accessible(n_j, n_i)*	$\tau_{v_a(e)}(n_i, n_j)$
	Over-all *End*	*connected(n_i, n_j)* *free(n_j)* *free(n_j)*	*End*	*train_at(t, n_j)* ¬ *free(n_j)* *engine_at(e, n_j)* *engine_damaged(t, n_i)* *track_accessible(n_j, n_i)* *track_accessible(n_j, n_i)*	
ResolveTrain Blockage*(t, n)*	*Start* *Over-all*	*train_blocked(t, n)* *train_at(t, n)* *train_at(t, n)*	*End*	¬ *train_blocked(t, n)*	$\rho(t, n)$
ClearBlocked Track*(n_i, n_j)*	*Start*	*track_blocked(n_i, n_j)*	*End*	*track_clear(n_i, n_j)* *track_clear(n_j, n_i)* ¬ *track_blocked(n_i, n_j)* ¬ *track_blocked(n_j, n_i)*	$\kappa(n_i, n_j)$

Note(s): Shorthands as in Table 2. The *DriveEngineToDamagedDownTrain* variant mirrors the up-variant, with *down* and *heading_down* in place of *up* and *heading_up*, and the engine heading flipped accordingly. Dispatch actions do not require the destination to be free, as it is occupied by the disabled train. $\mu(e)$, $\rho(t, n)$ and $\kappa(n_i, n_j)$ denote the times to attach the engine, resolve a train blockage and resolve a track disruption, respectively

specification presented in Section 4.2, with metric min total-time (makespan minimization). Table 6 presents the instance corresponding to the network of Figure 1.

Together, the domain model and problem instances encode railway scheduling and routing as a temporal planning task, from which temporal planners automatically generate timestamped action sequences that respect safety and gauge constraints, handle disruptions and minimize makespan in heterogeneous multi-gauge networks.

Table 5. Infrastructure-operation action schema

Action	Preconditions *ann.*	Preconditions *predicate*	Effects *ann.*	Effects *predicate*	Duration
Turnout *(n, n_1, n_2)*	*Start*	*junction_point(n)* *turnout_alternatives(n, n_1, n_2)* *connected(n, n_1)* $\neg$ *connected(n, n_2)*	*Start* *End*	$\neg$ *connected(n, n_2)* $\neg$ *connected(n_2, n)* *connected(n, n_2)* *connected(n_2, n)*	θ

Note(s): n_1, n_2 denote the alternative continuation points of the turnout at junction *n*, while the switching is in progress, neither alternative is connected

Table 6. Planning problem for the representative nominal instance of Figure 1, instantiating the objects, initial state (I_R) and goal (G_R)

Category		Predicates/values
Objects	Track points	*A, B, C, D, E, F*
	Trains	t_1 *(broad_gauge),* t_2 *(meter_gauge)*
	Stations	s_1
Initial state (I_R)	Deployment	*train_at*(t_1, *A*), *train_at*(t_2, *F)*
	Topology	*connected(A, B), connected(B, C), connected(B, D)* *connected(C, E), connected(D, E), connected(E, F)*
	Junctions	*junction_point(B), turnout_alternatives(B, C, D)* *junction_point(E), turnout_alternatives(E, C, D)*
	Gauges	*train_gauge*(t_1, *broad*), *train_gauge*(t_2, *meter)* *track_gauge(A, B, broad), ..., track_gauge(E,F, meter)*
	Services	*platform_at(C,* s_1*), platform_at(D,* s_1*),* *boarding_point*(t_1, *D*), *boarding_point*(t_2, *C)*
	Static values	$d(A, B) = 60$, $d(B, C) = d(B, D) = 10$, $d(C, E) = d(D, E) = 15$ $d(E, F) = 40$, $v(t_1) = 1.0$, $v(t_2) = 1.1$, $\beta(t_1, s_1) = 2$ $\beta(t_2, s_1) = 3$, $\theta = 1$
Goal (G_R)	Destinations	*train_at*(t_1, *F*), *train_at*(t_2, *A)*
	Services	*passengers_boarded*(t_1, *D*), *passengers_boarded*(t_2, *C)*
Metric		*Minimize total-time (makespan)*

Note(s): Symmetric *connected, track_accessible* and *track_gauge* facts

4.3 Temporal railway planning benchmark generation

Building on the proposed formulation, we constructed a benchmark comprising a single domain model and 200 problem instances, evenly divided into 100 nominal and 100 disrupted instances. Every instance models heterogeneous railway infrastructure with three track types (meter, broad and dual-gauge) and two train types (meter and broad-gauge) operating at different speeds, thereby capturing the gauge compatibility and speed heterogeneity of real railway systems. Table 7 summarizes the fixed parameters used during benchmark generation, including network characteristics, train movement properties, disruption settings and operational action durations. The benchmark generation procedures are presented in Algorithms 1 and 2.

GENERATENOMINAL function (Algorithm 1) takes the target numbers of trains, stations, track points and junctions together with a random seed as inputs. Then it constructs a connected railway network, assigns stations, junctions and trains, and generates the nominal planning task by producing the initial state I_R and goal state G_R. The generation process guarantees that every train has at least one gauge-compatible route between its origin and destination, ensuring that every nominal instance is solvable. Nominal-operation instances evaluate planning performance under normal operating conditions. Their complexity is

Table 7. Fixed parameters and sampling ranges used to generate benchmark instances across the experimental scales

Parameter	Value/range
Gauge types	{broad, meter}
Train speed (km/min)	$U(1.0, 1.2)$
Auxiliary engine speed (km/min)	1.0
Railway segment length (km)	$U(15, 35)$
Disrupted slowdown factor	$U(0.15, 0.30)$
Blocked track clear time (min)	$U(3, 60)$
Train blockage recovery time (min)	U(2, 60)
Turnout (track switching) time (min)	1
Boarding time (min)	$U(1, 5)$
Engine attach time (min)	$U(1, 15)$

Note(s): $U(a, b)$ denotes a continuous uniform distribution over *[a,b]*

determined solely by the railway infrastructure, with increasing numbers of trains, stations, track points and junctions.

Algorithm 1. Nominal railway instance generation.

1: **function** GENERATENOMINAL($n_{tr}, n_{st}, n_{pt}, n_{jc}, \sigma$)

Input: Number of trains (n_{tr}), stations (n_{st}), track points (n_{pt}), junctions (n_{jc}), random seed (σ)

Output: Nominal initial state I_R and goal G_R

2: Initialize an empty railway network using random seed σ.

3: Construct a connected dual-gauge corridor consisting of m track points.

4: Attach n_{st} stations to randomly selected corridor points using gauge-compatible tracks.

5: Insert n_{jc} junctions to create alternative parallel routes.

6: Assign n_{tr} trains to randomly selected gauge-compatible origin and destination stations.

7: Initialize track attributes and construct the connectivity and gauge relations.

8: Construct the nominal initial state I_R and goal state G_R.

9: **return** (I_R, G_R)

10: **end function**

On the other hand, GENERATEDISRUPTED (Algorithm 2) starts from a nominal instance and introduces the specified numbers of blocked trains, blocked track segments, slowdown-affected segments and engine failures to generate a disrupted initial state I'_R, while preserving the original goal state G_R. During disruption generation, track blockages are placed without disconnecting the railway network, ensuring that every disrupted instance remains solvable. Disrupted-operation instances evaluate planning performance under both increasing infrastructure size and disruption severity. The benchmark instances are organized into four primary scale categories: Small (S), Medium (M), Large (L) and Very Large (VL). Each category represents a progressive increase in infrastructure size and planning complexity. Within each category, multiple sub-levels (e.g. S1–S2, M1–M2, L1–L3 and VL1–VL3) provide finer-grained variations in problem difficulty. Tables 8 and 9 summarize the characteristics of the nominal and disrupted benchmark instances, respectively. The benchmark is designed with controlled and monotonic scaling. As the scale increases from S to VL, the numbers of trains, stations, track points and junctions increase consistently.

Table 8. Nominal-operation benchmark instances grouped by increasing problem size

Scale	Instances	Trains	Stations	Track points	Junctions
S1	p1–p10	3–12	5–17	50–127	2–7
S2	p11–p20	14–23	18–31	136–213	8–13
M1	p21–p30	25–35	32–45	222–300	14–20
M2	p31–p40	35–45	45–57	300–392	20–29
L1	p41–p50	46–56	59–71	402–494	30–39
L2	p51–p60	58–68	73–85	505–597	40–49
L3	p61–p70	69–80	87–100	607–700	50–60
VL1	p71–p80	80–92	100–115	700–793	60–72
VL2	p81–p90	93–106	117–132	803–896	73–86
VL3	p91–p100	107–120	134–150	906–1,000	87–100

Note(s): Instance complexity progressively increases with respect to the number of trains, stations, track points and junctions

Algorithm 2. Disrupted railway instance generation.

1: **function** GENERATEDISRUPTED ($I_R, n_{bt}, n_{btr}, n_{sd}, n_{ef}$)

Input: Nominal initial state (I_R), blocked trains (n_{bt}), blocked tracks (n_{btr}), slowdowns (n_{sd}), engine failures (n_{ef})

Output: Disrupted initial state I'_R

2: Select n_{bt} trains and assign blockage durations.

3: Introduce n_{btr} blocked track segments while preserving route connectivity for every train.

4: Apply speed reductions to n_{sd}.randomly selected unblocked track segments.

5: Introduce n_{ef} engine failures and assign a reachable rescue engine to each affected train.

6: Update the railway state with the generated disruptions.

7: **return** I'_R.

8: **end function**

For disrupted instances, disruption severity grows in parallel with infrastructure size through increasing numbers of blocked trains, blocked tracks, slowdown-affected segments, engine failures and auxiliary engines. This design enables systematic evaluation of planner scalability under progressively larger railway networks and increasingly challenging disruption scenarios.

5. Experimental results

This section presents a comprehensive experimental evaluation of the proposed temporal planning framework. We evaluate the railway domain and benchmark problem set using the temporal planners POPF (Coles *et al.*, 2010) and OPTIC (Benton *et al.*, 2012). POPF is a satisficing planner; it stops at the first valid solution. OPTIC extends POPF with anytime search that can keep improving a solution, but that mainly applies to preference- and cost-based objectives, which our domain does not use. Under plain makespan minimization here, OPTIC did not find a better solution than its first.

Consequently, all plan quality values reported in this section correspond to the quality of the generated (achieved) plans and should not be interpreted as optimal. To ensure correctness, we

Table 9. Disrupted-operation benchmark instances grouped by increasing problem and disruption complexity

		Problem characteristics				Disruption characteristics				
Scale	Instances	Trains	Stations	Track points	Junctions	Blocked train	Blocked track	Engine failure	Slow-down	Aux. eng.
S1	p101–p110	3–12	5–17	51–128	2–7	1–1	1–3	1–1	1–3	1–1
S2	p111–p120	14–23	18–31	137–214	8–13	1–1	3–4	1–1	3–5	1–1
M1	p121–p130	25–35	32–45	223–301	14–20	1–2	5–6	1–1	6–8	1–1
M2	p131–p140	35–45	45–57	302–395	20–29	2–3	10–13	2–3	13–17	2–3
L1	p141–p150	46–56	59–71	405–497	30–39	3–4	13–16	3–3	17–21	3–3
L2	p151–p160	58–68	73–85	508–601	40–49	4–5	16–19	3–4	22–26	3–4
L3	p161–p170	69–80	87–100	611–705	50–60	5–6	20–23	4–5	26–30	4–5
VL1	p171–p180	80–92	100–115	706–800	60–72	8–9	30–34	6–7	46–51	6–7
VL2	p181–p190	93–106	117–132	810–904	73–86	9–10	34–38	7–8	52–57	7–8
VL3	p191–p200	107–120	134–150	914–1,008	87–100	10–10	38–42	8–8	57–63	8–8

Note(s): In addition to infrastructure size, instances vary in disruption severity, including blocked trains, blocked tracks, engine failures, slowdown segments and available auxiliary engines

also validated all generated temporal plans using the VAL plan validator (Howey *et al.*, 2004), which verifies their semantic consistency with the PDDL domain specification.

The two planners (POPF and OPTIC) used in this experiment employ different search mechanisms and heuristic techniques for solving temporal planning problems. POPF uses grounded forward search with linear programming, and a Temporal Relaxed Planning Graph (TRPG) heuristic to guide exploration. OPTIC extends POPF with mixed-integer programming techniques for optimization, allowing preference-aware temporal planning and employs an enhanced TRPG heuristic with improved pruning during search. Both planners were executed with their default parameter settings and a maximum time limit of 30 minutes per problem instance. Since both planners operate deterministically, each problem instance was solved once. All experiments were conducted on a workstation equipped with a 5th-generation Intel Core i5 processor, and 8 GB RAM, running Ubuntu 24.04.3 LTS.

To evaluate the generated temporal plans, we consider several standard planning metrics: *makespan* (Definition 3), *plan length* and *plan generation time*. Makespan measures the total execution time of a plan, plan length denotes the total number of actions in the generated operational plan, and plan generation time is the wall-clock time required by a planner to produce a valid solution. Although these metrics assess the efficiency of the planning process, they do not directly capture the operational impact of disruptions on railway performance. We therefore complement them with domain-specific delay metrics that quantify the deviation of the generated schedules from ideal uninterrupted train operations.

5.1 Delay metrics

To assess the effectiveness of the proposed temporal planning model under realistic operational conditions, we analyze the delays incurred by trains during plan execution. Let Π_T denote a temporal plan, $T = 1, \ldots, n$ the set of trains, and $J_i = 1, \ldots, m_i$ the ordered sequence of track points traversed by train i in Π_T. We first define the overall schedule delay and then decompose it into domain-specific components corresponding to different disruption types. Total Delay (Definition 5) quantifies overall schedule degradation under operational conditions.

Definition 5. *(Total Delay).* The *total delay* of a temporal plan Π_T is the cumulative deviation between the actual and ideal arrival times of all trains at all track points:

$$TotalDelay(\Pi_T) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(T_{ij}^{actual} - T_{ij}^{ideal} \right),$$

where T_{ij}^{actual} and T_{ij}^{ideal} denote the actual and ideal arrival times of train i at track point j, respectively. The ideal arrival time assumes uninterrupted travel at nominal speed:

$$T_{ij}^{ideal} = T_{i0}^{dep} + \sum_{k=1}^{j} \tau_{ik}$$

where T_{i0}^{dep} is the departure time and τ_{ik} is the minimum traversal time of segment k.

Although total delay measures overall schedule degradation, it does not identify the underlying causes. We therefore decompose it into three components, each defined with respect to actions of our railway domain: slowdown delay (Definition 6), blockage delay (Definition 7) and engine failure delay (Definition 8).

Definition 6. The *slowdown delay* is the extra time incurred when trains traverse segments with reduced speed. For each drive-train or drive-assisted-train action on segment (n_i, n_j) let $\tau^{normal} = d_{n_i n_j}/s_i$ be the nominal duration and

$\tau^{slow} = d_{n_i n_j}/((1-\alpha_{n_i n_j})s_i)$ the actual duration under slowdown factor $\alpha_{n_i n_j} \in [0, 1)$, where $d_{n_i n_j}$ is the distance between points n_i and n_j, and s_i is the speed of train i. Then:

$$SlowdownDelay(\Pi_T) = \sum_{a \in \Pi_T \; a \in \{drive-train, drive-assisted-train\}} \left(dur(a) - \tau^{normal}\right),$$

where the difference is zero if $\alpha_{n_i n_j} = 0$.

Definition 7. The *blockage delay* is the total waiting time due to train or track blockages, computed as the cumulative duration of all resolve-train-blockage and clear-blocked-track actions:

$$BlockageDelay(\Pi_T) = \sum_{a \in \Pi_T \; a = resolve-train-blockage} dur(a) + \sum_{a \in \Pi_T \; a = clear-blocked-track} dur(a)$$

where each action duration is defined in the domain.

Definition 8. The *engine failure delay* is the total immobilization time of trains experiencing engine failure. For each train $i \in F$, recovery involves dispatching an auxiliary engine, attaching it and resuming movement. The delay is the sum of the dispatch and attach durations:

$$EngineFailureDelay(\Pi_T) = \sum_{i \in F} \left(dur\left(a_i^{dispatch}\right) + dur\left(a_i^{attach}\right)\right),$$

where durations are defined by the corresponding domain actions (drive-engine, drive-engine-to-damaged-train and attach-engine).

By decomposing total delay into slowdown, blockage and engine failure components with respect to our domain, this study provides a detailed assessment of how different types of disruptions affect the generated plans.

5.2 Overall experimental results

Table 10 summarize the performance of the OPTIC and POPF temporal planners on the proposed railway domain and benchmark problem set. Both planners solved all 100 nominal instances and 99 of the 100 disrupted ones. And produced identical makespan, plan length and delay values on every commonly solved instance; a single unified set of quality results is therefore reported. We attribute this convergence to OPTIC extending POPF with the same underlying TRPG heuristic and search strategy, and to the tightly constrained solution space imposed by gauge compatibility and single-track mutual exclusion, which leaves both planners little room to diverge in the plans they construct.

For nominal-operation instances, makespan remains stable across all scales with low standard deviations (34.2±1.9 minutes overall), plan length grows proportionally with network size and total delay is exactly zero, confirming conflict-free scheduling under undisrupted conditions. For disrupted-operation instances, makespan, plan length and delay all increase with problem scale. Plan-length growth reflects the added recovery operations, while total delay exhibits the most pronounced increase, driven by the escalating number and diversity of disruptions. The high standard deviations, particularly in delay, reflect inter-instance variability in disruption severity and rerouting availability, which we examine in Section 5.2.

The single failure, common to both planners, was the largest configuration in the benchmark (instance p200: 120 trains, 1,000 track points and 123 concurrent disruptions), on

Table 10. Experimental results of OPTIC and POPF on nominal and disrupted-operation instances

Scale	Nominal-operation instances Makespan	Plan length	Delay	Disrupted-operation instances Makespan	Plan length	Delay
S1	30.5 ± 3.1	15.0 ± 6.1	0.0	42.6 ± 19.4	18.0 ± 6.1	28.0 ± 42.2
S2	33.9 ± 1.4	37.0 ± 6.1	0.0	46.3 ± 18.8	40.0 ± 6.1	44.1 ± 92.8
M1	33.8 ± 0.8	59.2 ± 6.4	0.0	46.7 ± 25.4	62.7 ± 6.9	60.8 ± 140.9
M2	34.5 ± 1.3	79.6 ± 6.9	0.0	49.1 ± 14.5	86.7 ± 7.4	88.5 ± 176.5
L1	34.8 ± 1.5	102.4 ± 6.9	0.0	50.0 ± 25.1	112.2 ± 7.2	120.3 ± 229.6
L2	34.5 ± 0.6	125.8 ± 7.0	0.0	52.0 ± 20.2	138.2 ± 7.9	140.4 ± 254.5
L3	34.6 ± 0.9	148.6 ± 7.2	0.0	53.0 ± 24.0	163.3 ± 8.5	170.3 ± 317.3
VL1	35.0 ± 0.7	171.6 ± 8.3	0.0	54.2 ± 25.7	193.2 ± 9.4	217.1 ± 358.4
VL2	35.2 ± 1.0	199.0 ± 8.5	0.0	55.0 ± 29.6	224.1 ± 9.6	255.9 ± 425.3
VL3[a]	35.4 ± 0.7	226.6 ± 8.6	0.0	74.6 ± 16.2	251.7 ± 7.7	655.8 ± 490.4
Mean	*34.2 ± 1.9*	*116.5 ± 67.4*	*0.0*	*52.1 ± 22.8*	*127.7 ± 75.0*	*173.3 ± 319.8*

Note(s): [a]Disrupted VL3 statistics are computed over the nine solved instances
Makespan and delay are reported in minutes and plan length in actions, as mean ± standard deviation over ten instances per group. Both planners produced plans of identical quality on every commonly solved instance; a single unified set of quality metrics is therefore reported

which both planners exhausted the available physical memory of 8GB and were terminated by the operating system before reaching the 30-min time limit; no solution had been emitted at that point. Consequently, the disrupted VL3 statistics in Table 10 are computed over its nine solved instances.

All 199 generated plans were verified with the VAL validator (Howey *et al.*, 2004), which confirmed that every plan satisfies the temporal constraints, precondition requirements and goal conditions specified in the domain. No mutex violations, precondition failures, or goal-achievement errors were detected. This establishes the semantic correctness of the plans *with respect to the PDDL domain encoding*. Therefore, the VAL validation certifies the internal consistency of the generated recovery schedules for the encoded railway model, while deployment in practice would require an additional conformance layer between the planned schedule and the operational traffic-management system.

5.3 Delay analysis

No delay was observed on any nominal-operation instance, confirming that the generated schedules are conflict-free in the absence of disruptions. Figure 2 shows the average total delay

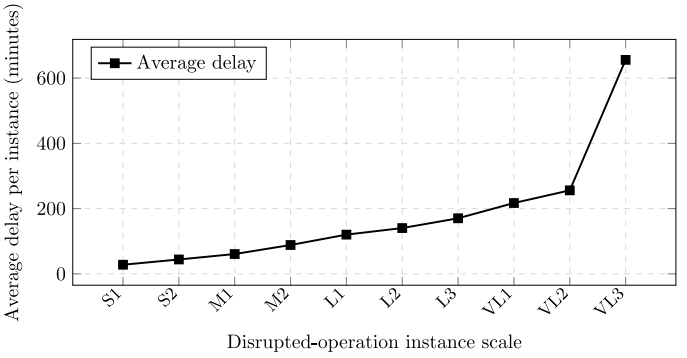

Figure 2. Average delay per instance for disrupted-operation instances (VL3 averaged over its nine solved instances). Both OPTIC and POPF generate identical coordination protocols with the same delay values. Delay grows near-linearly up to VL2 and surges at VL3, driven primarily by blockage resolution delay (cf. Figure 3)

across the disrupted-instance groups. Delay grows steadily and near-linearly with problem scale from S1 (28.0 minutes) to VL2 (255.9 minutes), and then rises sharply at VL3 (655.8 minutes over the nine solved instances). OPTIC and POPF produce identical delay values at every scale point, indicating that the domain is constrained tightly enough for both planners to converge on the same solution rather than finding different valid alternatives.

Figure 3 decomposes total delay by disruption type. Across all solved instances, slowdowns are the largest source of delay, accounting for 52.6% of total delay (9,029.7 minutes), followed by engine-failure recovery (28.1%, 4,813.0 minutes) and blockage resolution (19.3%, 3,313.0 minutes). Up to VL2, blockage delay remains the smallest component throughout (2.8–28.7 minutes per instance), indicating that the planners resolve track conflicts at modest temporal cost, typically by routing around affected segments, without violating temporal constraints. The VL3 surge, in contrast, is driven primarily by blockage-resolution delay, which increases 8.4× over VL2 (from 28.7 to 240.1 minutes per instance, 36.6% of VL3 delay) as the density of simultaneous blockages exhausts the available rerouting options and forces trains to wait for tracks to be cleared. Slowdown delay also more than doubles (2.2×, to 292.9 minutes per instance, 44.7% of VL3 delay). This distribution confirms that coordination overhead arises from operational constraints encoded in the domain rather than from modeling artifacts, and identifies rerouting capacity as the binding resource at the highest disruption densities.

Table 11 summarizes the delay distribution within each scale group: the first and third quartiles (Q1, Q3) bound the middle 50% of instances, and Max is the single worst instance in

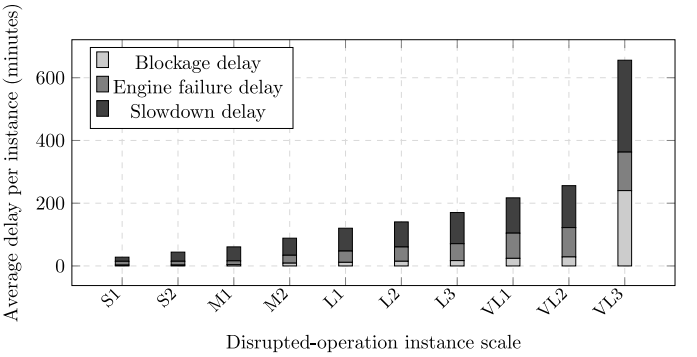

Figure 3. Breakdown of average delay per instance by disruption type across problem scale. Slowdown delay is the main contributor overall (52.6% of total delay), followed by engine-failure delay (28.1%) and blockage delay (19.3%)

Table 11. Distribution of total delay (minutes per instance) within each disrupted-operation scale group

Scale	Mean	Median	Q1	Q3	Max
S1	28.0	14.5	14.0	15.8	148.0
S2	44.1	15.0	14.2	15.0	308.1
M1	60.8	16.5	15.2	17.8	461.8
M2	88.5	32.5	32.0	33.8	590.8
L1	120.3	48.0	46.5	49.8	773.7
L2	140.4	62.5	60.5	64.8	864.6
L3	170.3	72.5	64.0	77.8	1073.2
VL1	217.1	106.5	99.8	108.5	1237.0
VL2	255.9	123.5	120.2	128.0	1466.3
VL3	655.8	481.0	356.0	575.0	1519.2
All	*173.3*	*62.0*	*17.5*	*115.0*	*1519.2*

the group. The distribution is strongly right-skewed (skewness 2.90 over all solved instances, overall median 62.0 minutes against a mean of 173.3). The inter-quartile ranges are narrow (e.g. L1: 46.5–49.8 min), yet one severely disrupted instance per group reaches 5 to 20 times the group median. In aggregate, the ten most delayed instances account for 58.9% of all delays in the benchmark, and the top five alone for 39.5%. Combined with the near scale-independence of slowdown delay (cf. Section 5.5, Figure 6), this shows that severe degradation depends on which segments are disrupted rather than on network size, while a few unfavorable ones where slowed or blocked segments lie on critical traffic paths dominate system-wide delay. This suggests that infrastructure hardening or recovery resources targeted at critical segments would yield disproportionate benefit.

5.4 Disruption resilience

Relating disrupted to nominal results quantifies how much of the incorporated disruption the generated recovery schedules absorb (Table 12). Three observations follow. First, the *makespan inflation factor*, the ratio of disrupted to nominal group makespan, stays between 1.37 and 1.56 from S1 through VL2, reaching 2.11 only at VL3. Second, the *recovery overhead*, additional actions in the disrupted plan relative to the nominal plan, remains within 6–13% of nominal plan length above S1, indicating that recovery relies on a modest number of targeted actions rather than rescheduling. Third, the *absorption ratio*, total train delay in train-minutes per minute of makespan extension, grows monotonically from 2.3 at S1 to 16.7 at VL3. Since total delay sums over all trains while makespan reflects only the longest completion, this growth indicates increasingly *parallel* absorption of disruption: trains incur delay simultaneously across separate parts of the network rather than serializing behind a single bottleneck. The recovery plans thus exploit the spatial parallelism of larger networks, the behavior required of a practical disruption-management tool.

5.5 Computational performance

Figures 4 and 5 show the growth of average computation time with problem size. Under nominal conditions (Figure 4), both planners scale smoothly and monotonically, from 0.09 to 0.12 seconds at S1 to 64.6 seconds (OPTIC) and 95.3 seconds (POPF) at VL3. For disrupted instances (Figure 5), computation times are substantially higher due to the additional coordination and recovery reasoning, and POPF's scaling exponent rises to 3.01, quantifying

Table 12. Resilience metrics per scale group, relating disrupted-operation results to nominal-operation baselines (group means)

Scale	Inflation	ΔMS (min)	Overhead (%)	Absorption
S1	1.39	12.0	20.0	2.3
S2	1.37	12.4	8.1	3.6
M1	1.38	12.9	5.9	4.7
M2	1.42	14.5	8.9	6.1
L1	1.44	15.2	9.6	7.9
L2	1.51	17.5	9.9	8.0
L3	1.53	18.4	9.9	9.2
VL1	1.55	19.3	12.6	11.3
VL2	1.56	19.8	12.6	12.9
VL3	2.11	39.2	10.8	16.7

Note(s): Δ MS is the increase in mean makespan over nominal, inflation is the disrupted/nominal makespan ratio, overhead is the additional plan length relative to nominal and absorption is mean total delay (train-minutes) per minute of makespan extension

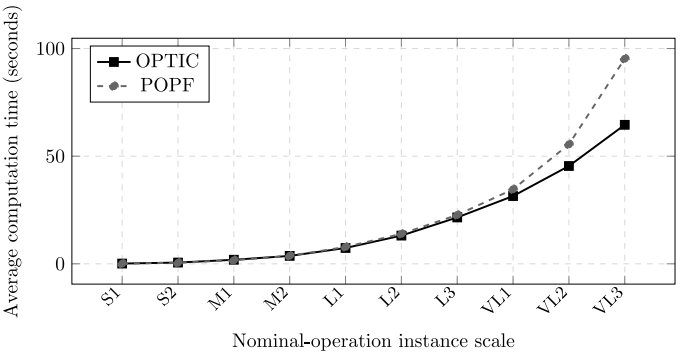

Figure 4. Average computation time per instance for nominal-operation instances. Both planners scale smoothly and monotonically, POPF is faster up to scale M1 and OPTIC from M2 onward, with the gap widening to 1.5× at VL3 (64.6 s against 95.3 s)

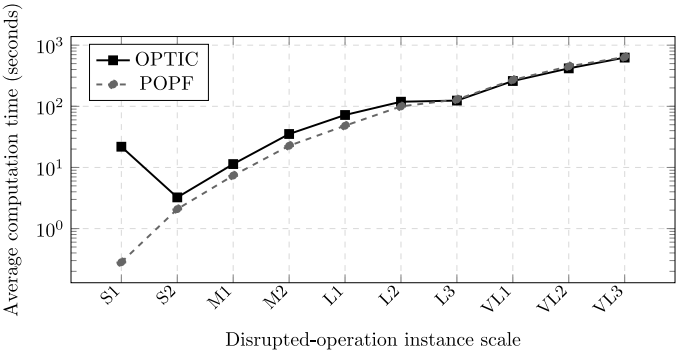

Figure 5. Average computation time per instance for disrupted-operation instances (VL3 averaged over its nine solved instances). POPF is faster on small instances (0.28 s at S1 against 21.83 s for OPTIC) but its runtime grows more steeply with scale

the extra computational burden that disruption handling imposes beyond network growth alone.

Even without disruptions, the two planners differ measurably in efficiency. OPTIC is faster overall on the nominal instance set (mean 19.0 s against 23.6 s, Wilcoxon signed-rank test $n = 100$; $z = -3.94$; $p < 0.001$). The same size dependence observed under disruption, at a smaller scale threshold: POPF is faster on 20 of the 21 smallest instances (plan length ≤50), whereas OPTIC is faster on 25 of 45 medium and 28 of 34 large instances. Their IPC agile scores are correspondingly close (97.47 for OPTIC against 96.54 for POPF), reflecting that neither planner dominates across the whole nominal benchmark.

Since both planners deliver identical plan quality, their comparison reduces to computational efficiency and robustness, summarized in Table 13. We assess efficiency with four measures. (1) The *Wilcoxon signed-rank test*, a non-parametric paired test applied to the per-instance runtime differences over the commonly solved instances. It makes no normality assumption, which is appropriate given the skewed runtime distribution. (2) The *IPC agile score*, the time-quality measure of the International Planning Competition: each instance contributes $1/(1 + log_{10}(t_i/t_i^*))$ to a planner's score, where t_i is that planner's runtime and t_i^* the fastest runtime recorded on instance i, unsolved instances contribute 0, so the maximum score equals the number of instances (here 100) and the fastest planner on an

Table 13. Computational comparison of OPTIC and POPF on the nominal and disrupted benchmark sets

Metric	Nominal OPTIC	POPF	Disrupted OPTIC	POPF
Coverage	100/100	100/100	99/100	99/100
IPC agile score	*97.47*	96.54	89.85	*98.04*
Geometric mean time (s)	*5.99*	6.16	45.59	*33.13*
Arithmetic mean time (s)	*18.97*	23.58	164.30	*162.14*
Faster, small instances (#)	1	*20*	0	*19*
Faster, medium instances (#)	*25*	20	8	*32*
Faster, larger instances (#)	*28*	6	*32*	7

Note(s): Size bands are defined by plan length (PL): small PL ≤ 50, medium $50 < \text{PL} \leq 150$, large PL > 150, band sizes are 21/45/34 instances (nominal) and 20/40/39 (disrupted)
Plan quality is identical for both planners on every commonly solved instance in both sets. All reported metrics are computed over the 199 solved instances (100 nominal, 99 disrupted)

instance receives the full point. (3) The *geometric mean* of runtimes, which, unlike the arithmetic mean, is not dominated by the largest instances when runtimes span four orders of magnitude. (4) The *runtime scaling exponent b*, obtained by least-squares regression of $log t$ on $log(plan\ length)$, i.e. $t \approx c \cdot PL^b$; throughout, plan length serves as the instance-scale proxy, since it grows deterministically with network size in the benchmark.

On the disrupted set, aggregate runtimes do not differ significantly (Wilcoxon, $n = 98$; $z = 0.87$; $p = 0.382$, OPTIC faster on 40 instances, POPF on 58, with one tie). POPF is faster on 19 of the 20 smallest instances (PL $\leq$50) and on 32 of the 40 medium instances ($50 < PL \leq 150$), which its higher agile score (98.04 against 89.85) and lower geometric-mean runtime (33.1 s against 45.6 s) reflect. OPTIC is faster on 32 of the 39 largest instances (PL > 150), so the ranking crosses over around scale L3. In terms of robustness, neither planner exhausted memory on any solved instance under the 8 GB budget: both terminated cleanly on all 100 nominal instances and all 99 solved disrupted instances. The sole memory failure across the entire benchmark occurred on the unsolved instance p200.

Relating planning cost to the operational timescale of the disruptions being managed, the longest single planning run across both planners and all solved instances was 795 s (13.3 min) on the disrupted set and 136 s on the nominal set. Mean planning time on disrupted instances ranges from 0.3 s at S1 to 10.7 min at VL3. Expressed against the delay each plan manages, the computational overhead of replanning amounts to between 0.02% (S1) and 2.9% (VL2) of the average disruption-induced delay per instance. Recovery-plan computation is therefore small relative to the operational timescale of the disruptions themselves, which supports the suitability of the framework for dispatcher-level disruption response. This conclusion is bounded by the benchmark: the unsolved instance p200 shows that the memory budget of the reported configuration is exhausted at the extreme end of the scale range.

5.6 Correlation analysis

To characterize how problem scale and disruption structure drive performance, we computed Spearman rank correlations at the level of individual instances ($n = 99$ commonly solved disrupted instances) presented in Figure 6. We use plan length as the problem-scale proxy, since it grows deterministically with the number of trains, track points and junction controllers in the benchmark design. Instance-level analysis is essential here: correlations computed on scale-group aggregates are trivially close to ± 1 for any quantity monotone in scale and would overstate the strength of the underlying relationships.

Three patterns emerge. First, computation time is driven almost entirely by problem scale ($\rho = 0.99$ with plan length for POPF, 0.95 for OPTIC) rather than by disruption severity, indicating that replanning cost in this domain is a function of network size, not of the amount of

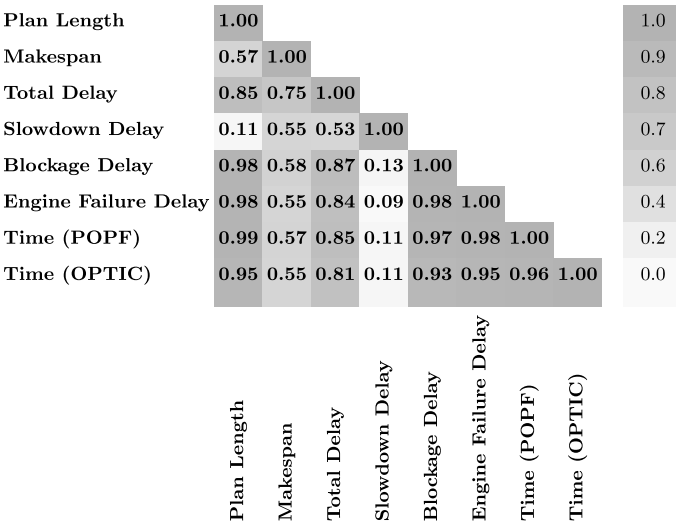

Figure 6. Spearman correlation heatmap over the $n = 99$ solved disrupted-operation instances. Coefficients with $|\rho| \geq 0.53$ are significant at $p < 10^{-9}$, while coefficients below 0.20 are not significant at the 0.05 level

disruption to be absorbed. Second, blockage and engine-repair delays correlate almost perfectly with scale ($\rho = 0.98$), consistent with the benchmark design in which the number of disruption events grows with network size. Third, and in contrast, slowdown delay is nearly scale-independent ($\rho = 0.11$, not significant) and instead tracks makespan ($\rho = 0.55$): the severity of slowdown-induced delay depends on which track segments are degraded and how central they are to the traffic flow, rather than on network size. Total delay correlates strongly with both scale ($\rho = 0.85$) and makespan ($\rho = 0.75$), confirming that the delay metrics capture genuine schedule degradation rather than an artifact of instance size alone.

6. Conclusion

In this study, a temporal planning-based framework for dynamic route optimization and disruption management in heterogeneous multi-gauge railway networks is developed. Our framework formulates railway operations in PDDL 2.1 to produce timestamped action plans with explicit constraints for gauge compatibility, operational safety and automated disruption resolution. An extensive evaluation on 200 benchmark instances shows that two temporal planners solve 99–100% of the instances with identical plan quality on commodity hardware, exhibit a size-dependent runtime crossover and generate recovery schedules whose delay decomposition and resilience characteristics scale consistently with network size and disruption intensity. The results confirm the potential of temporal planning to generate feasible and safety-compliant operational strategies even in large-scale and disruption-prone railway environments.

Overall, this work establishes temporal planning as a practical and scalable foundation for intelligent railway management systems. In future work, we intend to extend the framework along the improvement paths identified by our analysis: domain-specific heuristics and problem decomposition to push the observed scaling boundary, and plan repair and anytime replanning under uncertainty to handle incomplete or evolving operational information. Additional extensions, including derailment modeling and integration with real-time data streams, will further enhance the framework's resilience and deployment readiness.

Data availability

The planning domain, problem instances, results and validation outputs are available at https://github.com/PollobRay/Railway-Route-Planning.

References

Acuna-Agost, R., Michelon, P., Feillet, D., & Gueye, S. (2011). SAPI: Statistical analysis of propagation of incidents. A new approach for rescheduling trains after disruptions. *European Journal of Operational Research*, *215*(1), 227–243. doi: 10.1016/j.ejor.2011.05.047.

Albrecht, A. R., Panton, D. M., & Lee, D. H. (2013). Rescheduling rail networks with maintenance disruptions using problem space search. *Computers and Operations Research*, *40*(3), 703–712. doi: 10.1016/j.cor.2010.09.001.

Bangladesh Railway (2023). *Information book 2023*. Bangladesh Railway. Available from: https://railway.gov.bd/pages/files/6919975c933eb65569ddc4da

Benton, J., Coles, A., & Coles, A. (2012). Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, *22*(1), 2–10. doi: 10.1609/icaps.v22i1.13509.

Caff, S. C. M., Di Mauro, F., & Scala, E. (2014). A numeric PDDL based approach for temporally constrained journey problems. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence* (pp. 99–106). doi: 10.1109/ICTAI.2014.25.

Cardellini, M., Maratea, M., Vallati, M., Boleto, G., & Oneto, L. (2021). In-station train dispatching: A PDDL+ planning approach. In *Proceedings of the International Conference on Automated Planning and Scheduling*, *31*(1), 450–458. doi: 10.1609/icaps.v31i1.15991.

Cenamor, I., Vallati, M., & Chrpa, L. (2019). On the predictability of domain-independent temporal planners. *Computational Intelligence*, *35*(4), 745–773. doi: 10.1111/coin.12211.

Chai, S., Yin, J., D'Ariano, A., Liu, R., Yang, L., & Tang, T. (2024). A branch-and-cut algorithm for scheduling train platoons in urban rail networks. *Transportation Research Part B: Methodological*, *181*, 102891. doi: 10.1016/j.trb.2024.102891.

Coles, A., Coles, A., Fox, M., & Long, D. (2010). Forward-chaining partial-order planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, *20*(1), 42–49. doi: 10.1609/icaps.v20i1.13403.

Coviello, N., Medeossi, G., Nygreen, T., Pellegrini, P., & Rodriguez, J. (2023). Integrated ant colony optimization and mixed integer linear programming for multi-objective railway timetabling. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 4973–4979). doi: 10.1109/ITSC57777.2023.10421843.

Dündar, S., & Şahin, İ. (2013). Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. *Transportation Research Part C: Emerging Technologies*, *27*, 1–15. doi: 10.1016/j.trc.2012.11.001.

Fang, W., Yang, S., & Yao, X. (2015). A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems*, *16*(6), 2997–3016. doi: 10.1109/TITS.2015.2446985.

Fischetti, M., & Monaci, M. (2017). Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research*, *263*(1), 258–264. doi: 10.1016/j.ejor.2017.04.057.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, *20*(1), 61–124. doi: 10.1613/jair.1129.

Ghallab, M., Nau, D., & Traverso, P. (2004). Chapter 1 - introduction and overview. In M. Ghallab, D. Nau, & P. Traverso (Eds.), *Automated Planning* (pp. 1–16). Morgan Kaufmann. doi: 10.1016/B978-155860856-6/50004-1.

Gouveia, F., Albino, L., & Saldanha, R. L. (2025). A MaxSAT approach for the train timetabling problem with route choice and other features. In *Progress in Artificial Intelligence:*

24th EPIA Conference on Artificial Intelligence, EPIA 2025, Faro, Portugal, October 1–3, 2025, Proceedings, Part II (pp. 179–191). doi: 10.1007/978-3-032-05179-0_14.

Grigonis, V., Kaušylas, M., & Palevičius, V. (2025). Advancing sustainable interoperability between standard and broad-gauge railway systems. *Sustainability, 17*(18), 8336. doi: 10.3390/su17188336.

Haslum, P., Lipovetzky, N., Magazzeni, D., & Muise, C. (2019). Temporal planning. In *An Introduction to the Planning Domain Definition Language* (pp. 103–122). Springer International Publishing. doi: 10.1007/978-3-031-01584-7_5.

Howey, R., Long, D., & Fox, M. (2004). VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI '04* (pp. 294–301). doi: 10.1109/ICTAI.2004.120.

Ji, H., Wang, R., Zhang, C., Yin, J., Ma, L., & Yang, L. (2024). Optimization of train schedule with uncertain maintenance plans in high-speed railways: A stochastic programming approach. *Omega*, *124*, 102999. doi: 10.1016/j.omega.2023.102999.

Kang, L., Buhigiro, N., Sun, H., & Wu, J. (2024). A critical review of subway train timetabling and rescheduling problems. *IEEE Transactions on Intelligent Transportation Systems*, *25*(10), 12930–12942. doi: 10.1109/TITS.2024.3386728.

Lemos, A., Gouveia, F., Monteiro, P. T., & Lynce, I. (2024). Iterative train scheduling under disruption with maximum satisfiability. *Journal of Artificial Intelligence Research*, *79*, 1047–1090. doi: 10.1613/jair.1.14924.

Leutwiler, F., & Corman, F. (2023). A review of principles and methods to decompose large-scale railway scheduling problems. *EURO Journal on Transportation and Logistics*, *12*, 100107. doi: 10.1016/j.ejtl.2023.100107.

Li, D., Zhao, J., Peng, Q., Wang, D., & Zhong, Q. (2025). Rolling stock rescheduling in high-speed railway networks via a nested benders decomposition approach. *Transportation Research Part C: Emerging Technologies*, *171*, 105001. doi: 10.1016/j.trc.2025.105001.

Louadah, H., Papadakis, E., McCluskey, L., Tucker, G., Hughes, P., & Bevan, A. (2021). Translating ontological knowledge to PDDL to do planning in train depot management operations. In *Proceedings of PlanSIG 2021*. Available from: https://plansig2021.wordpress.com/

Narayanaswami, S., & Rangaraj, N. (2011). Scheduling and rescheduling of railway operations: A review and expository analysis. *Technology Operation Management*, *2*(2), 102–122. doi: 10.1007/s13727-012-0006-x.

National Academies of Sciences, Engineering, and Medicine (2007). *Rail freight solutions to roadway congestion: Final report and guidebook*. The National Academies Press. doi: 10.17226/14098.

Neves, G., Ribeiro, G., Grilo, M., Infante, V., & Andrade, A. R. (2024). Human reliability and organizational factors–how do human factors contribute to signals passed at danger? *Safety Science*, *171*, 106395. doi: 10.1016/j.ssci.2023.106395.

Nitisiri, K., Gen, M., & Ohwada, H. (2019). A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling. *Computers and Industrial Engineering*, *130*, 381–394. doi: 10.1016/j.cie.2019.02.035.

Nyporko, A., & Chrpa, L. (2025). Modelling and solving industrial production tasks as planning-scheduling tasks. *Data and Knowledge Engineering*, *157*, 102415. doi: 10.1016/j.datak.2025.102415.

Ul Islam, Z., Hossain Lipu, M. S., Karim, T. F., Fuad, A. M., Ali, M. M. N., Shihavuddin, A., & Al Mansur, A. (2024). Optimizing hybrid renewable energy based automated railway level crossing in Bangladesh: Techno-economic, emission and sensitivity analysis. *Energy Conversion and Management: X*, *24*, 100744. doi: 10.1016/j.ecmx.2024.100744.

Veelenturf, L. P., Kidd, M. P., Cacchiani, V., Kroon, L. G., & Toth, P. (2016). A railway timetable rescheduling approach for handling large-scale disruptions. *Transportation Science*, *50*(3), 841–862. doi: 10.1287/trsc.2015.0618.

Villalba Sanchis, I., Insa Franco, R., Martínez Fernández, P., & Salvador Zuriaga, P. (2021). Experimental and numerical investigations of dual gauge railway track behaviour. *Construction and Building Materials*, *299*, 123943. doi: 10.1016/j.conbuildmat.2021.123943.

Wang, Y., Song, R., He, S., & Song, Z. (2022). Train routing and track allocation optimization model of multi-station high-speed railway hub. *Sustainability*, *14*(12), 7292. doi: 10.3390/su14127292.

Xiu, C., Pan, J., D'Ariano, A., Zhan, S., & Peng, Q. (2024). Passenger service-oriented timetable rescheduling for large-scale disruptions in a railway network: A heuristic-based alternating direction method of multipliers. *Omega*, *125*, 103040. doi: 10.1016/j.omega.2024.103040.

Yan, F., Bešinović, N., & Goverde, R. M. P. (2019). Multi-objective periodic railway timetabling on dense heterogeneous railway corridors. *Transportation Research Part B: Methodological*, *125*, 52–75. doi: 10.1016/j.trb.2019.05.002.

Yang, L., & Liang, R. (2022). An AI planning approach to emergency material scheduling using numerical PDDL. In *Proceedings of the 2022 International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2022)* (pp. 47–54). doi: 10.2991/978-94-6463-010-7_7.

Zhang, H., & Ni, S. (2022). Train scheduling optimization for an urban rail transit line: A simulated-annealing algorithm using a large neighborhood search metaheuristic. *Journal of Advanced Transportation*, *2022*(1), 9604362. doi: 10.1155/2022/9604362.

Zhuo, S., Miao, J., Meng, L., Yang, L., & Shang, P. (2024). Demand-driven integrated train timetabling and rolling stock scheduling on urban rail transit line. *Transportmetrica A Transport Science*, *20*(3), 2181024. doi: 10.1080/23249935.2023.2181024.

Corresponding author
Sabah Binte Noor can be contacted at: sabah@duet.ac.bd