IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 31 October 2019; revised 13 August 2020; accepted 26 September 2020.
Date of publication 14 October 2020; date of current version 6 December 2021.

Digital Object Identifier 10.1109/TETC.2020.3030984

Parallel Computing for Multi-Objective Train Rescheduling

SAI PRASHANTH JOSYULA, JOHANNA TÖRNQUIST KRASEMANN, AND LARS LUNDBERG

The authors are with the Department of Computer Science, Blekinge Institute of Technology, 371 41, Karlskrona, Sweden
CORRESPONDING AUTHOR: S. P. JOSYULA (sai.prashanth.josyula@bth.se)

ABSTRACT In railway traffic systems, it is essential to achieve a high punctuality to satisfy the goals of the involved stakeholders. Thus, whenever disturbances occur, it is important to effectively reschedule trains while considering the perspectives of various stakeholders. This typically involves solving a multi-objective train rescheduling problem, which is much more complex than its single-objective counterpart. Solving such a problem in real time for practically relevant problem sizes is computationally challenging. The reason is that the rescheduling solution(s) of interest are dispersed across a large search tree. The tree needs to be navigated fast while pruning off branches leading to undesirable solutions and exploring branches leading to potentially desirable solutions. The use of parallel computing enables such a fast navigation of the tree. This article presents a heuristic parallel algorithm to solve the multi-objective train rescheduling problem. The parallel algorithm combines a depth-first search with simultaneous breadth-wise tree exploration while searching the tree for solutions. An existing parallel algorithm for single-objective train rescheduling has been redesigned, primarily, by (i) pruning based on multiple metrics, and (ii) maintaining a set of upper bounds. The redesign improved the quality of the obtained rescheduling solutions and showed better speedups for several disturbance scenarios.

INDEX TERMS Transportation, decision support, parallel algorithms, tree search strategies

I. INTRODUCTION

Public transportation is an important part of daily human life. Transportation by railways is one of the major components of public transportation. Disturbances in railway networks have a significant impact on the daily life of passengers. In railway disturbance management, there exist three major stakeholders: infrastructure managers, railway operators and passengers, each with diverse and potentially conflicting goals [1]. When rescheduling trains during a disturbance, the goals of an infrastructure manager are focused on the operational feasibility of the rescheduled timetable, while railway operators aim at minimizing operation costs [1]. The liberalization of the European railway sector in recent years has compelled railway operators to increase the focus on satisfaction of passengers [2]. Thus, it has become a crucial goal for the railway operating companies to strive for a better passenger satisfaction.

While solving a train rescheduling problem, it is frequently required to consider multiple, partially conflicting, objectives. In other words, it is usually required to solve it as a multi-objective optimization problem (MOP). Alternative objective functions may result in structurally quite different rescheduling solutions [3]. During train rescheduling, it is important to find a solution which is satisfactory from both a passenger-oriented perspective and an operational perspective. The former perspective takes into account the inconvenience caused to passengers while the latter takes into account operational feasibility, operational costs, etc.

By presenting a set of solutions with different tradeoffs between conflicting objectives to a human expert, one can provide valuable decision support in train rescheduling.

II. PROBLEM DESCRIPTION

In the railway sector, everyday train services are based on preplanned timetables which ensure feasibility of the services by respecting the applicable constraints. Typically, such constraints enforce safety by requiring a minimum time separation between consecutive trains passing through the same railway track. A railway disturbance is an unexpected event that makes the originally planned train timetable infeasible by introducing 'conflicts'. A conflict is a situation that arises when two trains require an infrastructure resource during overlapping time intervals, thus violating one or more constraints.

Disturbances are triggered by incidents such as overcrowded platforms, train malfunctions, signalling system failures, etc. Depending on the type of the incident and its severity, the induced delays are either minor or significant. Train timetables are planned with appropriate time margins in order to recover from minor delays. Hence, when a disturbance causes minor delays, the affected train(s) may be able to recover from the effects of the disturbance provided there is sufficient buffer in the original timetable. In case of a disturbance that causes a significant delay to one or more trains, conflicts arise in the existing timetable, making it operationally infeasible. The resolution of these conflicts in real time, to quickly obtain a feasible timetable (e.g., within 10 sec) of sufficient quality, constitutes train rescheduling.

In order to resolve a conflict, the following three tactics are frequently employed: (1) Reordering, i.e., prioritizing a train over another, (2) Retiming, i.e., allocating new arrival and departures times to one or more trains, (3) Local rerouting, i.e., reassigning tracks of one or more trains. Apart from these tactics, conflicts can also be handled by globally rerouting the trains, partially/fully cancelling the affected train services, etc. The algorithmic approach presented in this paper applies only reordering, retiming, and local rerouting.

During a disturbance, rescheduling the railway traffic is typically handled manually by train dispatchers who have very limited access to decision support systems [4]. The time available for analysing alternative decisions is often very limited. Under these circumstances, a rescheduling strategy often employed by train dispatchers is to prioritize the on-time trains over the trains that are delayed due to the disturbance. This strategy does not always lead to the best rescheduling solution as several potentially desirable alternative schedules are never considered. Thus, it is a challenge for the decision maker to analyze alternative desirable solutions and motivate his/her rescheduling choices within the available time. This challenge becomes even harder when the perspectives of multiple stakeholders need to be considered.

It is computationally difficult to reschedule the train traffic in real time while considering multiple objectives. Thus, it is challenging to find rescheduled timetables

1) that are of good quality, both from an operational as well as passenger-oriented perspective,
2) sufficiently fast, i.e., within the allowed computational time limit.

Balancing this tradeoff between speed and solution quality is a well-known challenge faced by current train rescheduling algorithms. Hence, there is a need to investigate faster solution approaches to train rescheduling that consider different perspectives. In the context of, e.g., a branch and bound (B&B) algorithm, considering multiple perspectives typically implies exploring larger portions of the search tree. The use of parallel algorithms enables exploring large search trees fast as compared to their sequential counterparts.

Recent advances in computer hardware have made powerful chips such as multi-core central processing units (CPUs) and graphics processing units (GPUs) increasingly common. In spite of this, limited research has been conducted in designing parallel algorithms that employ such hardware to better solve the train rescheduling problem. Recently, Josyula *et al.* [5] report significant speedup in train rescheduling as a result of parallel exploration of multiple branches of the search tree. Further research [6] in parallel algorithms explores the potential of GPUs in train rescheduling. However, these parallel algorithms have been employed in the context of a single-objective train rescheduling problem, the objective being *minimization of final delays of trains*. A recent review [7] of rail-research literature shows that passenger-oriented train rescheduling quite often needs to be posed as a multi-objective problem.

Thus, it is necessary to investigate the potential of parallel algorithms for multi-objective train rescheduling. Massive speedups reported for multi-objective B&B algorithms [8] make such an investigation a worthwhile endeavour. The aim of this research is to efficiently solve a multi-objective train rescheduling problem using a parallel algorithm.

This study investigates how a parallel heuristic search algorithm can be used to better solve a real-time railway rescheduling problem while considering perspectives of infrastructure managers and railway operators, as well as passengers.

In the context of a tree search algorithm, multiple perspectives can be considered by including multiple metrics for pruning (i.e., by relaxing the pruning). It is assumed that to better solve a rescheduling problem means:

1) to improve the quality of the obtained rescheduling solutions with respect to the considered evaluation metrics, and
2) to increase the computational speed of obtaining the rescheduled solutions.

The following propositions are formulated and investigated:

P_1: When pruning is based on multiple metrics, a larger number of solution branches of the search tree are explored. Thus, the risk of pruning branches leading to desirable solutions is reduced, which in turn may lead to finding potentially better solutions.

P_2: When pruning is based on multiple metrics, the speedup due to incorporating a parallel tree search algorithm is greater.

III. INTRODUCTION TO MULTI-OBJECTIVE OPTIMIZATION

Consider a MOP with i objectives, where each objective function f_i corresponding to the i^{th} objective needs to be minimized. A solution u is then said to *dominate* a solution v (denoted $u \prec v$) only if [9]:

1) u is at least as good as v in all of the objectives.
$$\forall i,\ f_i(u) \leq f_i(v).$$
2) u is better than v in at least one of the multiple objectives.
$$\exists i,\ f_i(u) < f_i(v).$$

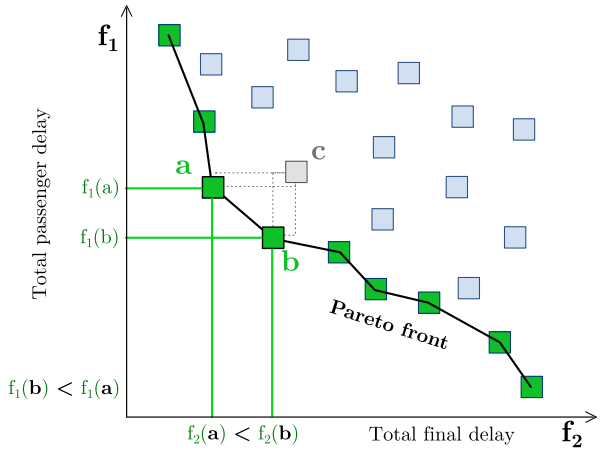


FIGURE 1. Illustration of a pareto front: Feasible solutions after train rescheduling.

Figure 1 illustrates, via an example, the concept of dominance in a bi-objective minimization problem in the context of real-time train rescheduling. As can be seen from the figure, both solution *a* and solution *b* dominate solution *c*.

A solution is said to be *pareto-optimal* (or *pareto efficient*) if it is not dominated by any other solution. A pareto-optimal solution cannot be improved in one objective without lowering the solution quality with respect to other objective(s) [9]. The set of all pareto-optimal solutions constitute the *pareto set* (also known as *pareto frontier* or *pareto front*). For every solution not in the pareto set, there exists at least one solution in the set that dominates it. The goal of multi-objective optimization [10] is two-fold:

1) to find a set of solutions as close as possible to the pareto front,
2) to find a set of solutions as diverse as possible.

In our case, this set of solutions is presented to a human expert who selects one of these solutions based on the situation at hand and his/her experience.

Example approaches that are often explored by the scientific community to solve MOPs are the ϵ-constraint method, the compromise programming method, etc. For a detailed introduction to multi-objective optimization, see [11, 12].

IV. RELATED WORK

The goals of a real-time train rescheduling algorithm are threefold [13]: (i) to quickly reach completion, (ii) to capably handle large, realistic input data, and (iii) to obtain high-quality solutions. Every algorithmic approach has its pros and cons; each approach differs in its ability to fulfill the above goals.

Multi-objective train (re)scheduling has been a topic of considerable research interest for many years. Recently, Binder *et al.* [2] solved a tri-objective railway rescheduling problem with special emphasis on minimizing passenger inconvenience. The problem is formulated as an integer linear program (ILP) that includes ϵ-constraints for two of the three objectives. For a realistic case study, they achieve high-quality solutions in terms of passenger satisfaction, with a minor increase in the timetable's operational cost. The drawback of their approach is that for many problem instances, it takes $>$ 1 hour to give solutions with an optimality gap of 3%. Due to this, they deem their approach impractical to use for real-world rescheduling.

More recently, Shakibayifar *et al.* [14] proposed a multi-objective version of the variable neighborhood search to solve the railway disturbances caused by a partial/full blockage. Their approach generates, for real-world test cases, sets of good quality solutions with minimized: (i) total average train delays at destinations, and (ii) deviation from the original timetable. However, the authors use a computational time limit of 15 min for solving each scenario, which is inapt for real-time applications.

In the context of train rescheduling, there is also significant research that focuses on incorporating multiple objectives using a single objective function. Examples are [15, 16], which (i) formulate the train rescheduling problem as a mixed-integer linear programming (MILP) model and an ILP respectively, (ii) consider two objectives: minimizing number of train cancellations, and train deviations/delays, (iii) use a single objective function that is the weighted sum of the considered two metrics. Apart from several drawbacks inherent to the weighted-sum approach [17], one of its practical difficulties is to come up with reasonable weights that are agreeable to the involved stakeholders, who may have conflicting goals.

Multi-objective B&B algorithms are widely employed in several application domains. Sourd and Spanjaard [18] present a formal framework to design such algorithms. Several surveys, each focusing on studies employing a specific type of algorithm to solve MOPs, exist in literature, e.g., B&B algorithm [19] and evolutionary algorithm [20]. Parallel computing paradigms are increasingly considered in the design and implementation of algorithms for MOPs [19].

Research studies that employ concepts of parallel computing in railway research have recently been reviewed in [21]. Though the use of parallel computing for railway rescheduling has been investigated in recent years, e.g., in [5, 6, 22], research on parallel computing for multi-objective train rescheduling is rather scarce. In a recent work, Nitisiri *et al.* [23] present a parallel multi-objective genetic algorithm for scheduling trains. Their algorithm employs a GPU and obtains a best-compromised solution. When considering two objectives, they obtain quick (in $<$ 1 sec) and promising results while scheduling trains on mass transit lines. The performance of the algorithm for (i) rescheduling on main lines, and for (ii) many (i.e., $>$ 3) objectives is unknown.

The research on reinforcement learning (RL) based approaches for train (re)scheduling is increasing in recent times [13, 24, 25]. Obara *et al.* [13] use a deep RL approach which effectively reschedules trains during disturbances caused due to train delays (of 5–35 min) on a small-scale problem (6 trains, 8 stations). Their approach, however, has difficulties in dealing with real-world input. Recently, Khadilkar [25] proposed a scalable RL algorithm for train

(re)scheduling, which completes its execution in 3–11 min for real-world problem instances (up to 440 trains, 60 stations). A noteworthy drawback in these approaches is that they use only one objective. Multiple objectives might need to be considered for obtaining higher quality solutions.

According to [19], most of the published multi-objective B&B algorithms are straight-forward extensions of the single-objective case and simply follow a depth-first search (DFS) strategy. Various other search strategies remain to be investigated [19]. Based on the review of related work, some of the observed weaknesses and challenges are addressed in this paper by: (i) investigating the effectiveness of a parallel search strategy, (ii) considering up to six relevant objectives, and (iii) quickly reaching completion for real-world input, while train rescheduling. This paper extends an existing single-objective train rescheduling algorithm [5] for the multi-objective case, using a few concepts from an algorithmic framework [18]. The redesigned algorithm is parallelized using a parallel DFS strategy. The main contributions of the research presented in this paper are: (i) findings from an experimental study showing the benefits of incorporating multiple objectives in train rescheduling, as well as identified challenges resulting from the expanded tree search, (ii) a proposed parallel algorithmic approach to address the above-mentioned challenges, (iii) a systematic assessment of quality-related properties of the resulting rescheduling solutions.

V. ALGORITHMIC DESIGN CHOICES

The sequential and the parallel heuristic algorithms for single-objective train rescheduling problems have been designed and introduced in [5]. The following provides a concise summary of the two algorithms.

The sequential algorithm constructs (and simultaneously navigates) a binary tree by iteratively detecting and resolving conflicts. Typically, a conflict can be between two or more trains. However, we chose a node to represent a conflict between exactly two trains. Starting with the root node, each node is visited using the depth-first search strategy to find the best solution. Throughout the search, the value of the upper bound is updated, based on which the branches leading to undesirable solutions are pruned. The root node corresponds to the original timetable which turns infeasible due to the disturbance. At each node, a conflict detection operation is performed on the corresponding timetable. The detected conflicts are arranged in a chronological order and the first conflict is chosen to be resolved. The outgoing edge corresponds to the rescheduling decision made as a part of conflict resolution. Reordering, retiming trains, and local rerouting are the employed rescheduling decisions. Leaf nodes in the unpruned branches correspond to feasible solutions.

Rescheduling Tactics. In any intermediate timetable state, i.e., at any internal node, first, one of the trains in the chosen conflict is prioritized over the other. Typically, each of the two outgoing edges corresponds to a prioritization alternative. Then, a child node is created by performing the following actions: (i) By locally rerouting the unprioritized train if an empty track is available throughout the train's occupancy of the conflict section, (ii) Otherwise, by making the unprioritized train wait on a prior section (likely causing a reordering), and retiming it accordingly to resolve the conflict. Thus, reordering is always accompanied by retiming. Each edge in the binary tree, i.e., each rescheduling decision, corresponds to either (i) a track reassignment, (ii) retiming, or (iii) reordering and retiming, of a train.

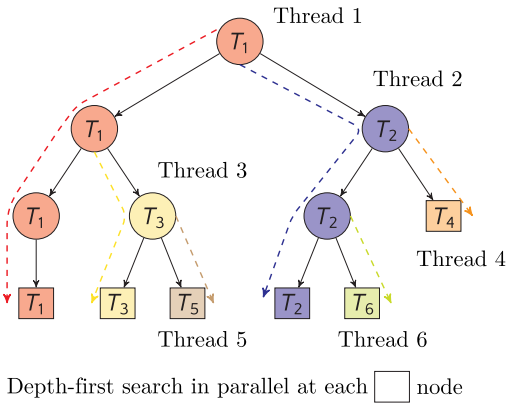


FIGURE 2. Illustration of parallel algorithm with 6 parallel threads.

The parallel algorithm decomposes the search tree construction into several disjoint tasks which can be computed in parallel. The algorithm takes as a parameter the maximum number of allowed parallel threads. During the execution of the program, once the specified number of threads are created, each thread runs in parallel an instance of the aforementioned sequential DFS algorithm with the appropriate node as its root node (illustrated in Figure 2). Throughout the execution of the parallel program, the operating system dynamically assigns threads to the available processors, and all the threads share and update the value of the upper bound. The best solution obtained by the parallel algorithm is the same as that obtained by the sequential algorithm. See [5] for further details about the two algorithms.

In the aforementioned algorithms, only a single evaluation metric (total final delay) is used for pruning. Incorporating several key evaluation metrics while pruning the search tree ensures that several perspectives can be considered in the computation of good-enough, or even optimal, solutions. Thus, in this study, the algorithms are redesigned such that several metrics can be considered while pruning. At the same time, instead of a single upper bound, a set of upper bounds associated with the set of best solutions is maintained. Each employed combination of pruning metrics corresponds to a multi-objective train rescheduling problem where the objectives are to minimize the values of the metrics used for pruning. Section V-B discusses, in detail, the modifications made to the aforementioned algorithms.

A. METRICS FOR SOLUTION EVALUATION AND PRUNING

The purpose of the evaluation metrics is to capture different effects of certain decisions and assess the properties (unwanted

as well as desired) of the alternative rescheduling solutions. Several comparable metrics exist for comparative evaluation of two railway timetables. In this section, the most relevant metrics for evaluation of rescheduled timetables, both from a passenger-oriented as well as an operational perspective, are presented. The decision maker, based on his/her experience and based on the evaluation metrics, selects the most desirable solution from the set of best solutions provided by the algorithm.

A partially rescheduled timetable is considered to be *undesirable* if further rescheduling will make it worse than one or more of the already available feasible rescheduled timetables. Intelligent navigation of the solution space requires discarding potentially undesirable solutions. In other words, while constructing a search tree, it is necessary to prune off branches which likely lead to undesirable solutions. The metrics chosen for evaluation are also used for pruning. Note that the minimization objectives of the MOP correspond to the chosen pruning metrics.

A positive deviation from the originally scheduled time in the initial timetable is called as a *delay*. Such a deviation in a train's timetable at its final station is called its *final delay*. The selected evaluation metrics are defined and the effect of using them for pruning is described as follows. For all the metrics, the lower the value of the metric, the better the quality of the rescheduling solution.

1) Total final delay (TFD): It is the sum of final delays of all the trains. An increase in the final delay lowers the quality of a solution. Hence, this metric is used for pruning off undesirable solution branches.
2) Total accumulated delay (TAD_2): It is the sum of delays ($>$ 2 min) in arrival times of all the trains at intermediary, scheduled commercial stations.[1] TFD does not consider what happens to the trains en-route, even though it is often partly reflected at the final destination [3]. Including this metric for pruning prevents discarding potentially desirable solutions with slightly higher TFD but with lower TAD_2.
3) Total passenger delay (TPD_2): It is the sum of delays ($>$ 2 min) incurred by alighting passengers, en-route as well as at the final destination. Including this metric for pruning prevents discarding potentially desirable solutions with, e.g., slightly higher TFD and, a lower TPD_2.
4) Number of delayed passengers ($\#D_2$pax): The number of passengers that experience a delay ($>$ 2 min) while alighting, i.e., while dismounting the train at their destination. Including this metric for pruning prevents discarding solutions with a higher value of TPD_2 but with a lower value of $\#D_2$pax.
5) Number of delayed trains (#Dtrains): The total number of trains that experience a final delay. Including this metric for pruning prevents discarding potentially desirable solutions with, e.g., slightly higher TFD and, a lower #Dtrains.
6) Number of trains with secondary delays ($\#D_2$sectr): The number of trains that at some point are recorded to have a delay ($>$ 2 min), excluding the trains that suffer from an initially forced delay due to the disturbance scenario. This metric considers the propagation of delays and monitors how initially punctual trains may be affected by already delayed trains. Including this metric for pruning retains solutions with lower values of $\#D_2$sectr even though they have higher values for other pruning metrics.

B. EMPLOYING MULTIPLE PRUNING METRICS

The heuristic rescheduling algorithms, originally introduced in [5], have been modified for multi-objective train rescheduling. These modifications are discussed as follows.

1) DEFINING DOMINANCE OPERATORS

Let $(u_1, u_2 \ldots u_n)$ and $(v_1, v_2 \ldots v_n)$ be the *cost-vectors* for any two solutions u and v. A cost-vector of a solution is defined as the n-tuple[2] representing the values of the n pruning metrics. Note that a solution can be either partial (i.e., at an intermediary node) or complete (i.e., at a leaf node). The following notations, similar to those defined in the multiobjective B&B framework by Sourd and Spanjaard [18], are adopted.

i) Weak dominance

$$u \preccurlyeq v \Leftrightarrow u_i \leq v_i, \forall\ i \in n, \tag{1}$$

ii) Dominance

$$u \prec v \Leftrightarrow u \preccurlyeq v \text{ and not } (v \preccurlyeq u). \tag{2}$$

2) MAINTAINING A SET OF UPPER BOUNDS

In the context of a single-objective optimization problem [5], during the navigation of solution space (or search tree), the upper bound UB is recorded in order to identify and prune undesirable branches. Typically, the value of this upper bound is the cost of the best feasible solution B found so far. Note that this is also the value of the objective to be minimized.

In solving the MOP under consideration, instead of a single best solution B and a corresponding upper bound UB, a set $\mathcal{B}$ of best solutions and a set $\mathcal{UB}$ of upper bounds (i.e., cost-vectors of solutions in $\mathcal{B}$) is maintained.

3) USING WEAK DOMINANCE OPERATOR TO PRUNE SOLUTION BRANCHES

While constructing and navigating the search tree to solve the multi-objective train rescheduling problem, the concept of weak dominance is used at every node. Its use can be classified into the following two cases:

I. At an intermediary node, if there exists a solution in $\mathcal{B}$ that weakly dominates the partial solution corresponding

[1]A commercial station is a station where a train is not allowed to leave before its originally scheduled departure time.

[2]an n-tuple is an ordered sequence of n elements, usually written by listing the elements within ( ) and separated by commas.

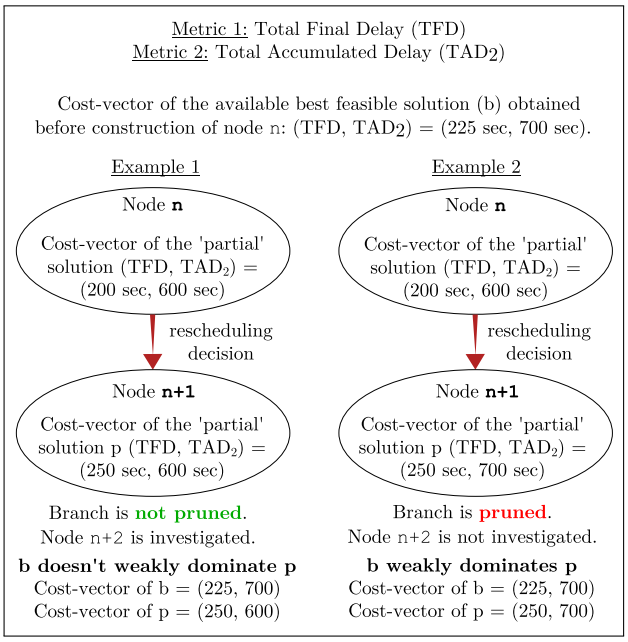


FIGURE 3. Examples to illustrate pruning while solving a bi-objective problem where the objective is to find a solution that minimizes both TFD and TAD_2.

to the node, then the branch is pruned. Otherwise, the branch is explored further.

II. At a leaf node, as mentioned in the multi-objective B&B framework [18],
   a) If none of the solutions in $\mathcal{B}$ weakly dominate the obtained solution b_{new}, then b_{new} is included in $\mathcal{B}$.
   b) All the solutions $b_i \in \mathcal{B}$ such that $b_{new} \prec b_i$ are removed from $\mathcal{B}$ once b_{new} is inserted.

The use of the weak dominance operator ($\preccurlyeq$) has the following two consequences corresponding to the cases outlined above:

I. Assume that during the construction of a solution branch, at an intermediary node, the corresponding partial solution is p. If a solution b in the then available list of best solutions $\mathcal{B}$ has the same cost-vector as p, then the solution branch is pruned.

   Note that if the $\prec$ operator were used for pruning (instead of the $\preccurlyeq$ operator), the solution branch of p would not be pruned.

II. Assume that after constructing a solution branch, at the leaf node, the corresponding solution is b_{new}. If b_{new} has the same cost-vector as a solution b in the then available list of best solutions $\mathcal{B}$, then b_{new} is discarded.

Figure 3 illustrates the pruning through relevant examples. In both the examples in the figure, at the time of constructing node n of the tree, the set $\mathcal{B}$ of available best solutions = $\{b\}$, corresponding to which, $\mathcal{UB} = \{(225, 700)\}$. In Example 1 of the figure, in case of the partial solution p corresponding to node n+1, $b \preccurlyeq p$ is not true. Thus, the feasible solution obtained by navigating along the branch has the potential to be included in the set $\mathcal{B}$. Hence, navigation along the branch is continued.

Algorithm 1. The Algorithm for Multi-Objective Train Rescheduling (an abridged version)

Input: Original timetable, infrastructure, disturbance, multiple objectives.
Output: Set $\mathcal{B}$ of best, feasible solutions.

```
1: Update the timetable as per the disturbance.
2: Use the time supplements available in the timetable (for
   further details, see [5]).
3: Configure the ≼ operator based on the multiple objectives.
4: Create the empty set B of best solutions.
5: Construct (Root node).

6: Function Construct (node):
7:   Detect conflicts in the partial timetable p.
8:   if no conflict is detected then   /* Leaf node */
9:     Save the feasible timetable.
10:    Update the solution set (p), return.
11:  else if ∃b ∈ B : b≼p then return.   /* Prune */
12:  else   /* Construct the child nodes */
13:    for edge = "left edge", "right edge" do
14:      Resolve conflict (edge)
         Construct (child node).
15:      Restore the state of the parent node.
16:    end
17: return
18: Function Update the solution set (b_new):
19:  if ∄b ∈ B such that b≼b_new then
20:    foreach b ∈ B such that b_new ≺ b do exclude b
21:    Include b_new in the set B.
22: return
23: Function Resolve conflict (edge):
24:  Select the 'earliest' conflict from the detected ones.
25:  if the edge is "left edge" then prioritize a train
26:  else prioritize the other train
27:  Resolve the selected conflict using the appropriate
     rescheduling tactic (see Section V, for further details
     see [5]).
28: return
```

In Example 2 of the figure, in case of the partial solution p corresponding to node n+1, $b \preccurlyeq p$ is true. Thus, owing to the properties of the search tree under consideration, any feasible solution that will be obtained by further pursuing along this branch will be weakly dominated by solution b. Hence, the branch is pruned.

Algorithm 1 is an abridged version of the designed multi-objective algorithm. This algorithm is parallelized as explained in Section V and as shown in Figure 2. In the full version of the algorithm, whenever the parent node's conflict is between trains in the same direction, only the first train that enters the conflict section is prioritized (see Lines 13–15, 25–26 of Algorithm 1).

VI. EXPERIMENTAL DESIGN

In this study, an experiment is designed, based on the guide to experimental algorithmics by McGeoch [26]. The formulated propositions are revisited while highlighting the portions that are of relevance in this section.

TABLE 1. Algorithm parameters in the experiment.

Parameter	Scale
Nature of the algorithm	Categorical: {Sequential, Parallel}
Number of threads	Numerical: $\{1, 2, 3, \ldots\}$
Pruning criterion and corresponding objectives	Categorical: $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{63}\}$

TABLE 2. Considered pruning criteria and their respective pruning metrics.

Criterion	Pruning metrics used in the criterion
$\mathcal{P}_1$	TFD
$\mathcal{P}_2$	TFD, TAD_2
$\mathcal{P}_3$	TFD, TAD_2, TPD_2
$\mathcal{P}_4$	TFD, TAD_2, TPD_2, #D_2pax
$\mathcal{P}_5$	TFD, TAD_2, TPD_2, #D_2pax, #Dtrains
$\mathcal{P}_6$	TFD, TAD_2, TPD_2, #D_2pax, #Dtrains, #D_2sectr

P_1**:** When pruning is based on multiple metrics, a larger number of solution branches... are explored. Thus... lead to finding potentially better solutions.

P_2**:** When pruning is based on multiple metrics, the speedup due to incorporating a parallel tree search algorithm is greater.

A. PARAMETERS

The parameters of the heuristic rescheduling algorithm are mentioned in Table 1. The program used to reschedule can be run sequentially or in parallel. When run sequentially, the number of threads used to explore the search tree is equal to 1. However, when run in parallel, the number of threads is a parameter that can take the value of any positive integer.

Relaxation of pruning is achieved by changing the pruning criterion $\mathcal{P}_i$. The appropriate pruning criterion increases or reduces (i.e., restricts or relaxes) the pruning of branches in the search tree by considering fewer or more pruning metrics respectively.

Josyula *et al.* [5] solve a single-objective train rescheduling problem using the minimization of TFD as the objective and thus TFD as the pruning metric. The 6 pruning metrics (outlined in Section V-A) provide $2^6 - 1 = 63$ choices to select from. Out of these, the pruning criteria $\mathcal{P}_1$–$\mathcal{P}_6$ (each corresponding to their own set of minimization objectives) are chosen for the experiment. Table 2 lists the chosen pruning criteria and the pruning metrics comprising each criterion.

Table 4 presents the scenarios that comprise the representative sample of disturbances used for the experiment. This sample was originally defined and used for the experiment in [5]. In each disturbance scenario, a selected train suffers a delay at a selected section. The algorithm can also solve other types of disturbance scenarios, e.g., those caused due to a malfunctioned train or an infrastructure failure. However, in this experiment, we do not evaluate the algorithm for these types of scenarios.

TABLE 3. Factors and levels used in the experiment.

Factor	Level
Type of the heuristic algorithm	(Sequential, Parallel)
Number of threads	(1, 64)
Pruning criterion	$(\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_6)$
Disturbance scenario number	$(1, 2, \ldots 40)$

The timetable and the infrastructure data used in the experiment can be classified as *real instances*. The railway network is from Karlskrona–Tjörnarp. The infrastructure consists of 59 sections (including stations), and all tracks are bi-directional. The original timetable is from 15:50 to 21:10 (5 hr 20 min). The passenger data consists of the number of passengers alighting a train at each station, and is generated by a random number generator in C++. The number of alighting passengers (i) at any station are ≤ 18, (ii) at any commercial station are ≥ 5.

In the experiment, we assumed all trains as passenger trains with passengers alighting at all stations. An advantage of this assumption is that the individual trains have little influence on the rescheduling algorithm when $\mathcal{P}_3$–$\mathcal{P}_6$ are used as the pruning criteria. In practice, non-commercial stations are not used for passenger alighting and boarding, but merely as, e.g., train meeting or overtaking points.

B. PERFORMANCE INDICATORS

In the experiment, the following dimensions of algorithm performance are measured:

1) Explored solution branches, solution set quality.
   The quality of the set of solutions obtained from applying the rescheduling algorithm is measured as follows. The rescheduling solutions are (i) visualized and inspected using a train timetable visualization tool, (ii) compared using the chosen evaluation metrics.
2) Increase in speedup.
   The percentage increase in speedup of the parallel algorithm when employing a relaxed pruning criterion is measured.

For each disturbance scenario, for a run of the algorithm (corresponding to a criterion $\mathcal{P}_i$), the number of explored branches is denoted by $N(\mathcal{P}_i)$. This value represents the number of alternative solutions investigated before reaching at the solution set comprising the rescheduling solutions. We define $\%N(\mathcal{P}_{ij})$ as the percentage increase in the number of explored branches when changing from criterion $\mathcal{P}_i$ to $\mathcal{P}_j$ (see Equation (3)).

$$\%N(\mathcal{P}_{ij}) = \frac{N(\mathcal{P}_j) - N(\mathcal{P}_i)}{N(\mathcal{P}_i)} \times 100. \tag{3}$$

The speedup $S(\mathcal{P}_i)$ compares the speed of the parallel algorithm with respect to the sequential algorithm.

$$S(\mathcal{P}_i) = \frac{\text{Sequential algorithm run time}}{\text{Parallel algorithm run time}} = \frac{t_{seq} \text{ for } \mathcal{P}_i}{t_{par} \text{ for } \mathcal{P}_i}. \tag{4}$$

TABLE 4. Disturbance scenarios and results.

Scenario	Description of the disturbance scenario		Criterion $\mathcal{P}_1$			Seq. alg.		Average run-time of 5 trials (sec)			
Nr#	Disturbed train, delay time, section	Time Window	Obtained soln.	Optimal soln.	Difference (Rel. gap %)	$N(\mathcal{P}_1)$	$\%N(\mathcal{P}_{16})$	$Seq(\mathcal{P}_1)$	$Seq(\mathcal{P}_6)$	$Par_4(\mathcal{P}_1)$	$Par_4(\mathcal{P}_6)$
1	Train 1076 delayed 5 min at VÖV ÖND1	2.67 hrs	0.00 min	0.00 min	0.00 (0%)	25	436%	0.06	0.07	0.07	0.07
2	Train 1097 delayed 5 min at GUA NÄT	4.20 hrs	0.00 min	0.00 min	0.00 (0%)	1068	172%	0.14	0.49	0.07	0.07
3	Train 1250 delayed 5 min at VÖV ÖND1	3.37 hrs	12.65 min	8.03 min	4.62 (58%)	570	158%	0.08	0.23	0.08	0.12
4	Train 1267 delayed 5 min at CR	2.03 hrs	1.50 min	1.23 min	0.27 (22%)	16	0%	0.08	0.09	0.06	0.06
5	Train 1846 delayed 5 min at VÖV ÖND1	4.97 hrs	2.13 min	1.95 min	0.18 (9%)	20	0%	0.09	0.09	0.08	0.08
6	Train 1977 delayed 5 min at SAK_L3 SÖG	3.71 hrs	2.80 min	2.80 min	0.00 (0%)	8	0%	0.07	0.09	0.06	0.07
7	Train 1978 delayed 5 min at BML SÖG	2.73 hrs	3.48 min	2.52 min	0.96 (38%)	4	25%	0.06	0.06	0.06	0.06
8	Train 6175 delayed 5 min at CR1 KAP	3.11 hrs	10.32 min	0.00 min	10.32 (-)	2170	18%	0.04	0.4	0.06	0.24
9	Train 1076 delayed 13 min at VÖV ÖND1	2.67 hrs	3.57 min	2.65 min	0.92 (35%)	44	1805%	0.09	0.18	0.07	0.14
10	Train 1097 delayed 13 min at GUA NÄT	4.20 hrs	16.10 min	15.73 min	0.37 (2%)	51044	443%	4.53	40.71	0.07	0.28
11	Train 1250 delayed 13 min at VÖV ÖND1	3.37 hrs	26.77 min	24.53 min	2.24 (9%)	1336	2112%	0.21	3.75	0.08	0.24
12	Train 1267 delayed 13 min at CR	2.03 hrs	9.50 min	9.23 min	0.27 (3%)	22	214%	0.06	0.06	0.06	0.08
13	Train 1846 delayed 13 min at VÖV ÖND1	4.97 hrs	18.05 min	17.35 min	0.70 (4%)	388	123%	0.14	0.38	0.23	0.27
14	Train 1977 delayed 13 min at SAK_L3 SÖG	3.71 hrs	21.23 min	18.80 min	2.43 (13%)	289	26%	0.17	0.19	0.09	0.09
15	Train 1978 delayed 13 min at BML SÖG	2.73 hrs	18.80 min	16.87 min	1.93 (11%)	10	430%	0.1	0.1	0.07	0.07
16	Train 6175 delayed 13 min at CR1 KAP	3.11 hrs	26.48 min	9.20 min	17.3 (188%)	1947	7354%	0.27	16.9	0.12	1.35
17	Train 1076 delayed 17 min at VÖV ÖND1	2.67 hrs	17.72 min	16.80 min	0.92 (5%)	-	-	-	-	0.23	0.65
18	Train 1097 delayed 17 min at GUA NÄT	4.20 hrs	21.97 min	18.22 min	3.75 (21%)	98095	272%	7.4	52.75	0.51	4.53
19	Train 1250 delayed 17 min at VÖV ÖND1	3.37 hrs	32.03 min	32.03 min	0.00 (0%)	10	0%	0.06	0.06	0.06	0.06
20	Train 1267 delayed 17 min at CR	2.03 hrs	15.52 min	13.63 min	1.89 (14%)	196	0%	0.08	0.08	0.06	0.06
21	Train 1846 delayed 17 min at VÖV ÖND1	4.97 hrs	30.60 min	28.68 min	1.92 (7%)	76703	845%	7.95	114.82	0.07	2.56
22	Train 1977 delayed 17 min at SAK_L3 SÖG	3.71 hrs	26.80 min	26.80 min	0.00 (0%)	9	0%	0.07	0.08	0.06	0.06
23	Train 1978 delayed 17 min at BML SÖG	2.73 hrs	27.52 min	24.87 min	2.65 (11%)	35	86%	0.06	0.06	0.07	0.07
24	Train 6175 delayed 17 min at CR1 KAP	3.11 hrs	35.62 min	20.82 min	14.80 (71%)	12025	10037%	2.18	150.52	0.19	2.87
25	Train 1076 delayed 21 min at VÖV ÖND1	2.67 hrs	40.52 min	37.00 min	3.52 (10%)	13269	140%	1.59	3.66	0.07	2.60
26	Train 1097 delayed 21 min at GUA NÄT	4.20 hrs	24.10 min	22.73 min	1.37 (6%)	53336	6083%	7.40	488.59	0.09	5.5
27	Train 1250 delayed 21 min at VÖV ÖND1	3.37 hrs	45.87 min	41.07 min	4.80 (12%)	1081	1612%	0.19	2.81	0.38	1.03
28	Train 1267 delayed 21 min at CR	2.03 hrs	19.15 min	18.85 min	0.30 (2%)	62	16%	0.06	0.07	0.07	0.08
29	Train 1846 delayed 21 min at VÖV ÖND1	4.97 hrs	43.73 min	35.20 min	8.53 (24%)	103917	707%	7.47	83.45	0.87	59.75
30	Train 1977 delayed 21 min at SAK_L3 SÖG	3.71 hrs	34.80 min	34.80 min	0.00 (0%)	23	474%	0.07	0.08	0.07	0.08
31	Train 1978 delayed 21 min at BML SÖG	2.73 hrs	40.53 min	33.30 min	7.23 (22%)	204	994%	0.08	0.3	0.06	0.06
32	Train 6175 delayed 21 min at CR1 KAP	3.11 hrs	44.92 min	38.52 min	6.40 (17%)	13917	-	1.72	> 900	0.12	> 900
33	Train 1076 delayed 25 min at VÖV ÖND1	2.67 hrs	55.60 min	52.48 min	3.12 (6%)	3452	258%	0.57	1.79	0.2	0.65
34	Train 1097 delayed 25 min at GUA NÄT	4.20 hrs	36.45 min	33.08 min	3.37 (10%)	133704	79%	9.98	36.27	0.08	0.72
35	Train 1250 delayed 25 min at VÖV ÖND1	3.37 hrs	63.30 min	59.08 min	4.22 (7%)	-	-	-	-	0.17	-
36	Train 1267 delayed 25 min at CR	2.03 hrs	26.80 min	24.23 min	2.57 (11%)	65	9%	0.06	0.06	0.07	0.07
37	Train 1846 delayed 25 min at VÖV ÖND1	4.97 hrs	59.53 min	49.23 min	10.30 (21%)	79881	-	7.92	> 900	1.82	> 900
38	Train 1977 delayed 25 min at SAK_L3 SÖG	3.71 hrs	48.08 min	47.42 min	0.66 (1%)	1271	1561%	0.21	2.63	0.07	0.16
39	Train 1978 delayed 25 min at BML SÖG	2.73 hrs	44.53 min	43.57 min	0.96 (2%)	12	42%	0.06	0.06	0.06	0.06
40	Train 6175 delayed 25 min at CR1 KAP	3.11 hrs	68.28 min	53.18 min	15.10 (28%)	499938	-	22.9	> 900	0.41	52.16
	Average values of sequential and parallel implementations (on 4 cores):						1044%	2.22	28.63	0.18	3.81

The percentage increase in speedup $\%S(\mathcal{P}_{ij})$ is computed as follows.

$$\%S(\mathcal{P}_{ij}) = \frac{S(\mathcal{P}_j) - S(\mathcal{P}_i)}{S(\mathcal{P}_i)} \times 100. \tag{5}$$

Theoretically, for a given input scenario, with a relaxed pruning criterion, the algorithm takes longer to reach completion. For example, t_{seq} for $\mathcal{P}_6 \geq t_{seq}$ for $\mathcal{P}_1$, and t_{par} for $\mathcal{P}_6 \geq t_{par}$ for $\mathcal{P}_1$. Note that for any input scenario, when the values of t_{seq} and t_{par} are recorded through the experiment, the resulting value of $\%S(\mathcal{P}_{ij})$ can be negative as well.

C. FACTORS, LEVELS, DESIGN POINTS AND TRIALS

A factor is a parameter that is explicitly manipulated in the experiment. A level is a value assigned to a factor in an experiment. Table 3 lists the factors and the levels assigned to them. A design point is a particular combination of levels that are tested in the experiment. In this experiment, not all combinations of the levels listed in Table 3 are tested.

The sequential program uses one thread to explore the branches of the tree sequentially, in a depth-first manner. For each disturbance scenario, the pruning criteria $\mathcal{P}_i$ where $i = \{1, 2 \ldots 6\}$ are employed and the 40 scenarios (see Table 4) are solved. This gives rise to $1 \times 6 \times 40 = 240$ design points. For the 40 design points corresponding to $\mathcal{P}_1$, the performance indicator $\%N(\mathcal{P}_{16})$ is recorded. These values serve to investigate Proposition 1.

The parallel program is also used to solve the 40 disturbance scenarios in Table 4. The number of threads that explore multiple branches of the tree in parallel is set to 64. Only the pruning criteria $\mathcal{P}_1$ and $\mathcal{P}_6$ are employed. This gives rise to $1 \times 2 \times 40 = 80$ design points. For the 40 design points corresponding to $\mathcal{P}_1$, the performance indicator $\%S(\mathcal{P}_{16})$ is recorded. These values serve to investigate Proposition 2.

A single run of the program at a specific design point, which produces a measurement of the performance indicator is called as a *trial* or a *test* [26]. At each design point, 5 trials are conducted.

D. IMPLEMENTATION AND PLATFORM DETAILS

The train rescheduling algorithms, originally devised in [5], have been implemented in C++. These C++ implementations of the sequential and parallel algorithms are improved and

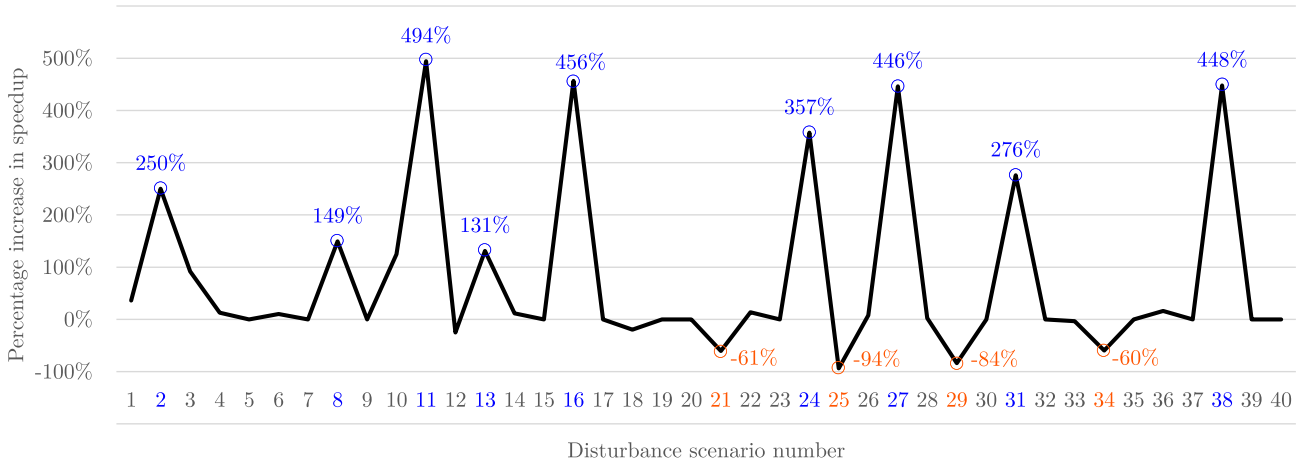


FIGURE 4. Percentage increase in speedup: $\%S(\mathcal{P}_{16})$, when pruning criterion is changed from $\mathcal{P}_1$ to $\mathcal{P}_6$ (see Equation (5)).

reused for the experiment in this study. Noteworthy improvements are discussed as follows.

The implementations of the algorithms in [5] construct a tree data structure (using the Boost Graph Library 1.64) while searching the solution space. This data structure stores rescheduling details related to the tree nodes and edges. These details are crucial for the visualization of the decisions taken by the algorithms. In practice, building such a data structure is meaningful as intelligent decision support systems require information visualization. However, in the implementations used in this study, tree data structures are not built during solution space navigation.

The implementations of algorithms in [5] use the dominance operator to prune solution branches. In this study, the weak dominance operator is used for pruning, as explained in Section V-B3.

The experiment is performed on a laptop equipped with a quad-core CPU (Intel Core i7-8550U). The available random-access memory is 16 GB. The underlying operating system is Windows 10 Education, and the compiler used to compile the C++ code is Microsoft C/C++ Optimizing Compiler Version 19.14.26431 for x64.

VII. RESULTS AND ANALYSES

The results of the experiment are presented in Table 4. For the sake of brevity, the values of $N(\mathcal{P}_i)$ for $i > 1$ are not presented. For scenarios 17, 32, 35, 37 and 40, the execution times could not be recorded as either (i) the algorithm could not reach completion even within 15 min, or (ii) the recursive implementation of DFS led to the overflow of the call stack. The optimal solutions for single-objective rescheduling (i.e., using $\mathcal{P}_1$) are obtained by means of an MILP model outlined in [27].

When the pruning criterion is relaxed from $\mathcal{P}_i$ to $\mathcal{P}_j$, where $i,j \in \{1,2,3,\ldots 6\}$ and $i < j$, the following is observed. Typically, the sequential as well as the parallel algorithm take more time to reach completion. When the pruning criterion is changed from $\mathcal{P}_1$ to $\mathcal{P}_6$, the average execution time of the sequential algorithm increased from 2.22 sec to 28.63 sec, while that of the parallel algorithm increased from 0.18 sec to 3.81 sec. Note that, on average, both the algorithms execute in less than 30 sec (also, see Table 4).

The reason for the increased execution times is that a larger portion of the search tree is explored, while retaining branches that were otherwise pruned when employing the stricter criterion $\mathcal{P}_1$. While employing $\mathcal{P}_6$, the average increase in the number of solution branches explored by the sequential algorithm is 1044%. Results show that this increase in the explored solutions is typically associated with an improved solution set that has additional better solutions.

A. IMPROVEMENT IN SPEEDUP

Table 4 shows the execution times of the algorithms for criteria $\mathcal{P}_1$ and $\mathcal{P}_6$ for the 40 disturbance scenarios. Figure 4 shows the increase in speedups across the scenarios when the train rescheduling algorithm is run in parallel on a quad-core computer. These values are computed using Equations (4) and (5). The average speedups for the single-objective and multi-objective train rescheduling are $\frac{2.22}{0.18} \approx 12$ and $\frac{28.63}{3.81} \approx 8$ respectively. Though the former speedup is greater than the latter, for some disturbance scenarios, the speedup attained with $\mathcal{P}_6$ is greater than that attained with the use of $\mathcal{P}_1$ (see Figure 4, Table 4).

The results in Figure 4 show that the percentage increase in speedup $\%S(\mathcal{P}_{16})$ can be greater than 350%, for time-consuming[3] disturbances (e.g., scenarios 10, 24). For few disturbance scenarios, changing the pruning criterion from $\mathcal{P}_1$ to $\mathcal{P}_6$ decreased the speedup attained by the use of parallel algorithm. However, such a decrease in speedup is of less importance. The reason is that, despite a decrease in speedup, the parallel algorithm still runs very fast. For example, see scenario 21 in Table 4. Even though the value of $\%S(\mathcal{P}_{16}) = -61\%$, the time taken by the parallel algorithm is 2.56 sec, compared to 114.82 sec taken by the sequential algorithm. The increase in speedup achieved for other scenarios is

[3]scenarios that are time-consuming to solve using the sequential algorithm with $\mathcal{P}_6$.

TABLE 5. Execution times of parallel algorithm for two time-consuming scenarios.

Disturbance scenario	Criterion	Explored branches	Execution time	
		Seq	*Seq*	Par_4
24	$\mathcal{P}_1$	12,025	2.18	0.19
	$\mathcal{P}_6$	1,218,979	150.52	2.87
26	$\mathcal{P}_1$	53,336	7.4	0.09
	$\mathcal{P}_6$	3,297,678	488.59	5.5

TABLE 6. Additional solutions obtained with each pruning criterion (Scenario 3).

$\mathcal{P}_i$	TFD	TAD_2	TPD_2	$\#D_2$pax	#Dtrains	$\#D_2$sectr
$\mathcal{P}_1$	12.7 min	24.3 min	9.8 hr	135	3	4
$\mathcal{P}_3$	15.7 min	33.3 min	9.1 hr	152	4	5
	21 min	30.1 min	9.5 hr	131	4	4
$\mathcal{P}_4$	20 min	34.5 min	9.4 hr	135	4	5
	25.2 min	31.3 min	9.8 hr	114	4	4
	22.3 min	32.5 min	12.9 hr	117	3	4
	22 min	51 min	12.9 hr	115	3	3
	26 min	62.5 min	15.1 hr	107	3	2
	21.3 min	51 min	13.5 hr	129	3	4

significant, both for time-consuming scenarios (e.g., scenario 24) and for easier scenarios (e.g., scenario 38).

From the aforementioned results, the following is observed. When the number of explored branches grows large, the distribution of the desirable solution(s) in the search tree is a property that can significantly affect (i.e., increase) the execution time of the sequential DFS algorithm. In contrast, the execution time of the parallel search algorithm is often only slightly affected by this property. For example, see the scenarios in Table 5.

B. DISCUSSIONS ON SELECTED SCENARIOS

A detailed discussion of the results obtained for selected disturbance scenarios are as follows.

Scenario 3. In the solution obtained by the use of pruning criterion $\mathcal{P}_1$ (see Table 6), along with the originally disturbed Train 1,250, two other trains experience a final delay of approximately 4 minutes each, totalling to 12.7 minutes. When the pruning criterion is changed to $\mathcal{P}_2$, more branches of the search tree are explored, but the final solution set remains the same, i.e., no new solutions are obtained.

When $\mathcal{P}_3$ is used as the pruning criterion, two additional solutions with lower TPD_2 (of 9.1 hr, 9.5 hr) are obtained (see Table 6). This reduction in TPD_2 is achieved at the cost of increasing both TFD and TAD_2. Also, for one of the solutions, a reduction in TPD_2 is achieved at the cost of increased number of delayed passengers. Table 7 presents the metrics related to some of the rescheduled trains in the solution with TFD = 21 min.

Using pruning criterion $\mathcal{P}_4$, a higher number of additional solutions are obtained. However, for most of these solutions, along with a decrease in the number of delayed passengers, a significant increase in TAD_2 and TPD_2 can be noticed (see Table 6). Using $\mathcal{P}_5$ or $\mathcal{P}_6$, no additional solutions are obtained.

TABLE 7. Metrics related to few rescheduled trains for Scenario 3, $\mathcal{P}_3$.

Train	TFD	TAD_2	TPD_2	$\#D_2$pax
Train 1250	4.9 min	4.4 min	1.9 hr	24
Train 1095	6.5 min	0	1.5 hr	15
Train 1263	5.8 min	11.7 min	2.4 hr	25
Train 1857	3.8 min	8.3 min	1.9 hr	28

Note: Due to the random passenger flow distribution, we have 15 passengers of Train 1095 alighting at a non-commercial station, thus giving a non-zero TPD_2 and $\#D_2$pax, even though the TAD_2 is 0.

1) IMPROVEMENT IN PUNCTUALITY AT COMMERCIAL STATIONS

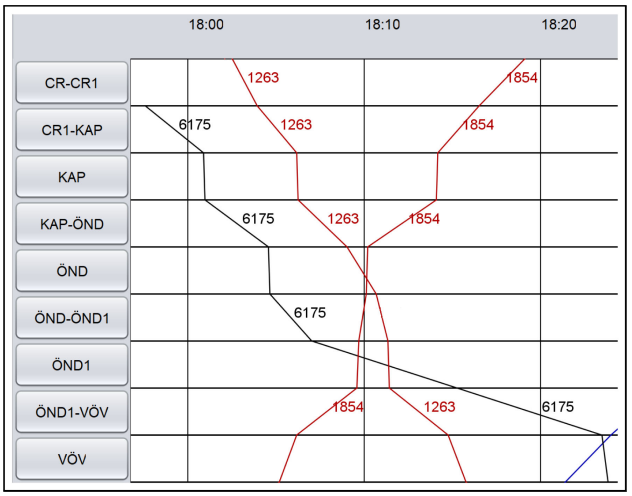
Scenario 8. Figure 5 shows a small portion of the rescheduling solution obtained by the use of pruning criterion $\mathcal{P}_1$. In this solution, the initially disturbed Train 6175 does not experience a delay at its final station. This train gains significant time at the station labelled ÖND1 (see Figure 5(b)). However, Train 1263 experiences a delay of 10.3 min at its final station. This train has a TAD_2 of 39.4 min and 66 passengers experience a delay of $>$ 2 min (in total, 8.3 hr) while alighting. Though the Train 6175 reaches all its commercial stations as well as its final destination without any delay, it experiences delays at a few intermediary stations, thus causing 14 passengers to experience a delay of $>$ 2 min (in total, 1.1 hr) while alighting.

Apart from these trains, two other trains undergo platform track reassignments.

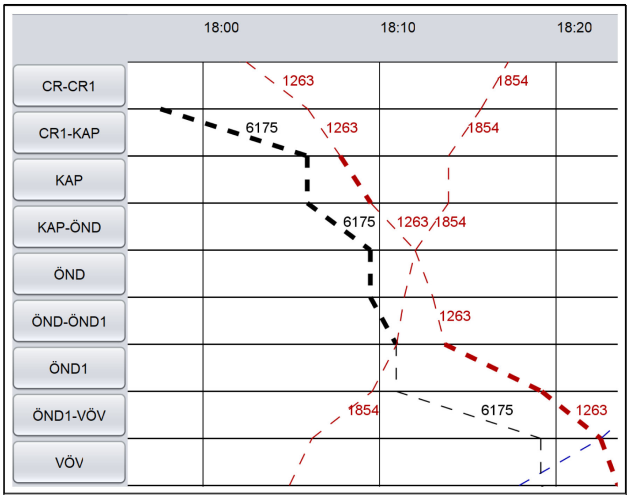
When the pruning criterion is changed to $\mathcal{P}_2$, the rescheduled trains are more punctual at commercial stops compared to the previous solution (see Table 8). At final stations, the disturbed train experiences a delay of around 2 min, while two other trains experience a delay of around 5 min (see Table 9). Compared to the previous solution, the punctuality of Train 1263 at commercial stops is increased, since TAD_2 reduced from 39.4 min to 27.8 min. This solution is a good alternative to the previously obtained solution.

The significant increase in the TPD_2 is shown in Figure 6. The number of passengers alighting each station is shown in the figure. It can be seen how the value of TPD_2 can easily increase even though another metric improves.

When the pruning criterion is changed to $\mathcal{P}_3$, an additional solution is obtained in which two trains experience a final delay of 37 sec and 14 sec. Ignoring these minor delays, the obtained solution has a (TFD, TPD_2) = (13.9 min, 12.1 hr). Compared to the previous solution's (13 min, 12.3 hr), for an increase in TFD of 1 minute, the TPD_2 can be reduced by around 0.2 hr. From a passenger perspective, this solution is certainly of a better quality compared to the previously obtained solution. Using pruning criterion $\mathcal{P}_4$, a solution



(a) Original timetable. The solid lines are the initial scheduled train paths.



(b) Rescheduled timetable for $\mathcal{P}_1$. The dotted lines are the paths after rescheduling. Significant delays as a result of rescheduling are shown in bold dotted lines.

FIGURE 5. A portion of original and rescheduled timetables for scenario 8, $\mathcal{P}_1$.

TABLE 8. Additional solutions obtained with each pruning criterion (Scenario 8).

$\mathcal{P}_i$	TFD	TAD_2	TPD_2	#D_2pax	#Dtrains	#D_2sectr
1	10.3 min	39.4 min	9.4 hr	80	1	1
2	13 min	27.8 min	12.3 hr	129	3	2
3	13.9 min	27.8 min	12.1 hr	129	5	2
4	18.1 min	27.8 min	13.3 hr	110	4	2

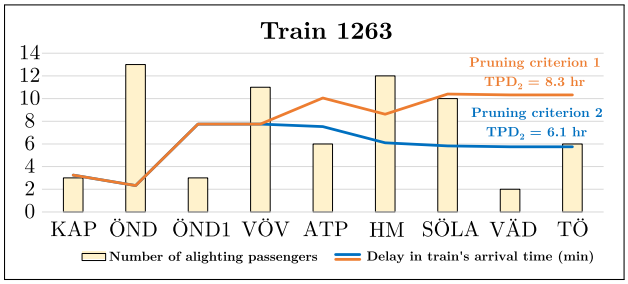
with (TAD_2, #D_2pax) = (27.8 min, 110) is obtained. The use of pruning criteria $\mathcal{P}_5$ and $\mathcal{P}_6$ did not give any new solutions.

2) BETTER SOLUTION QUALITY FROM A PASSENGER PERSPECTIVE

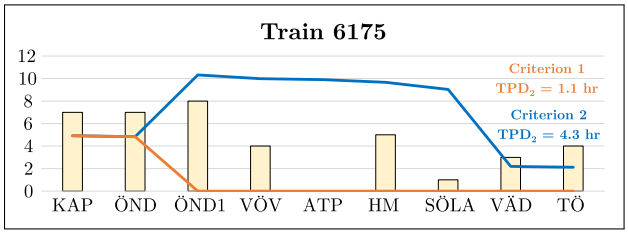
The use of pruning criterion $\mathcal{P}_6$ often produces one or more additional solutions of significantly better quality compared to

TABLE 9. Metrics related to few rescheduled trains for Scenario 8, $\mathcal{P}_2$.

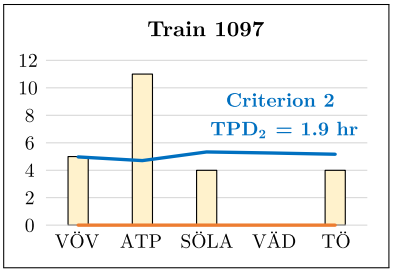
Train	TFD	TAD_2	TPD_2	#D_2pax
Train 6175	2.1 min	0	4.3 hr	39
Train 1097	5.2 min	0	1.9 hr	24
Train 1263	5.7 min	27.8 min	6.1 hr	66



(a) Delays experienced by Train 1263 at stations.



(b) Delays experienced by Train 6175 at stations.

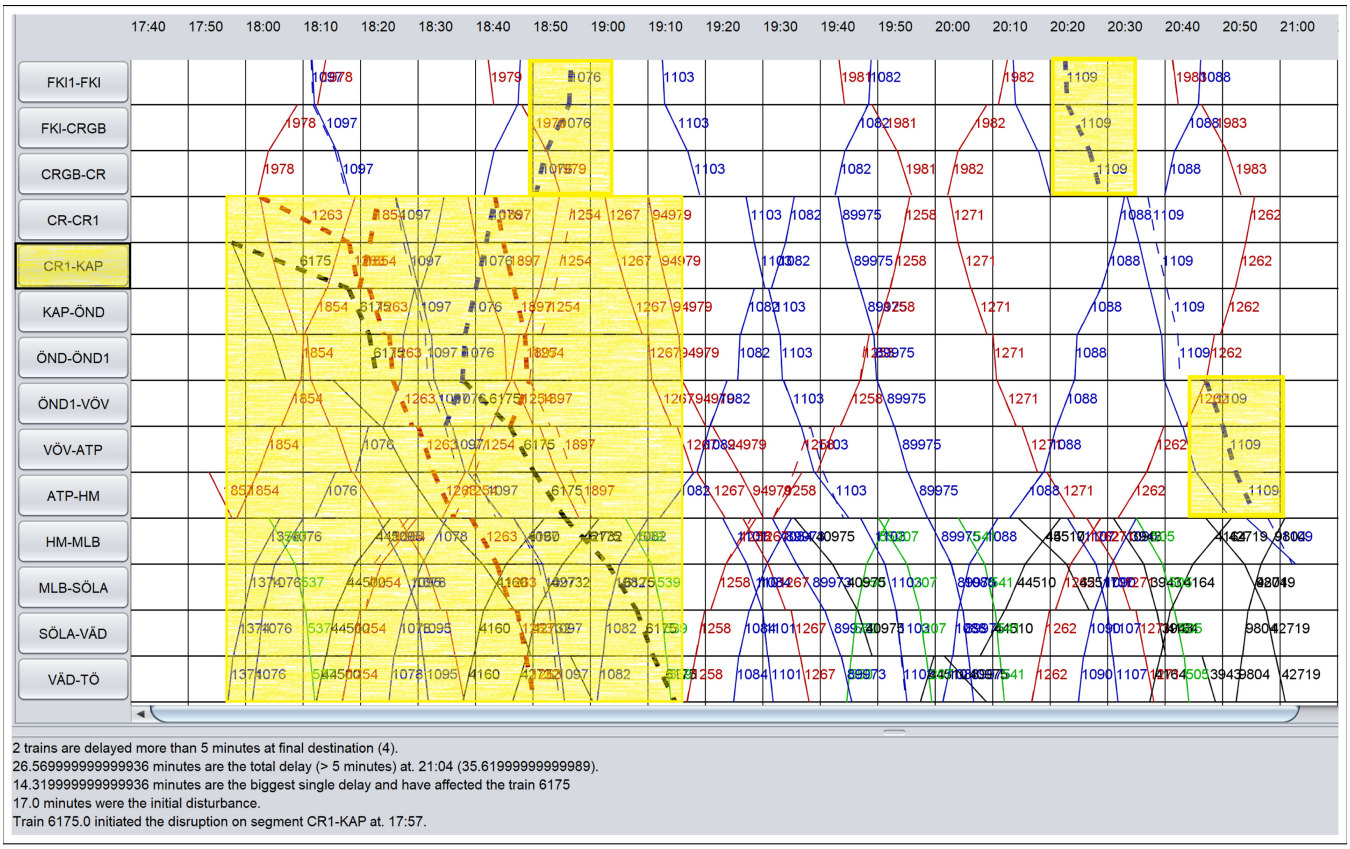


(c) Delays experienced by Train 1097 at stations.

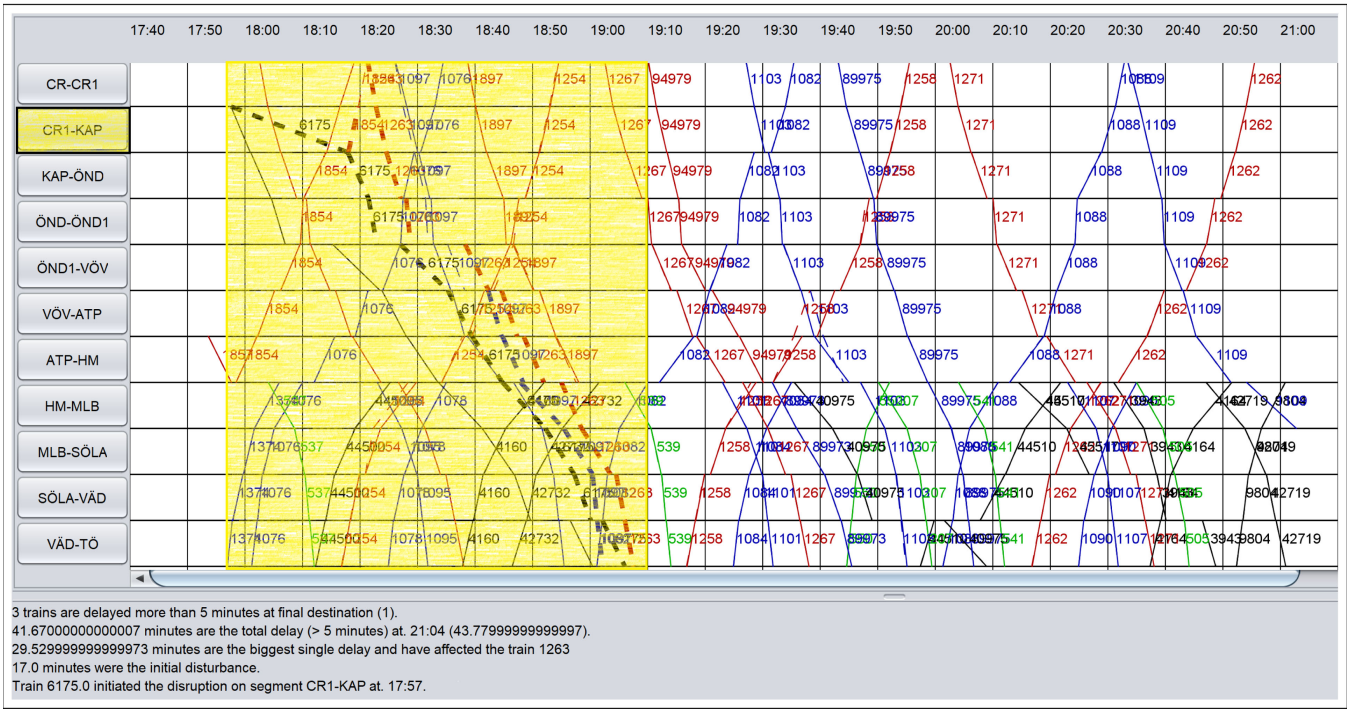
FIGURE 6. Delays experienced by few trains at their stations in solutions obtained with $\mathcal{P}_1$ = 1 and 2 (see Table 8).

$\mathcal{P}_1$. An example is Scenario 24, which is discussed as follows. In this scenario, Train 6175 experiences an initial delay of 17 min at section CR1-KAP. In the rescheduling solution obtained with the use of $\mathcal{P}_1$ (see Figure 7(a)), Train 1076 is rescheduled early on in its itinerary. As a consequence of rescheduling Train 1076, Train 1109 is also rescheduled quite early in its itinerary (not shown in Figure 7) to avoid conflicts. These rescheduling decisions cause delays at several stations along the single-tracked line. As a result, the number of delayed passengers and the accumulated passenger delay are quite high (400 passengers and 58.7 hr respectively).

In a rescheduling solution obtained with the use of $\mathcal{P}_6$, though the total final delay increases by 8.2 minutes, the TAD_2 is reduced by 8.9 minutes. The particular advantage of this solution is that the rescheduling is limited to a



(a) A rescheduled timetable obtained from using $\mathcal{P}_1$.



(b) A rescheduled timetable from using $\mathcal{P}_6$.

FIGURE 7. A portion of rescheduled timetables for Scenario 24. The solid lines are the initial scheduled train paths. The dotted lines are the paths after rescheduling. Significant delays as a result of rescheduling are shown in bold dotted lines. For the on-time trains, the solid and the dotted lines overlap.

TABLE 10. A solution obtained with use of $\mathcal{P}_6$ (Scenario 24).

$\mathcal{P}_i$	TFD	TAD_2	TPD_2	$\#D_2pax$	#Dtrains	$\#D_2sectr$
$\mathcal{P}_1$	35.6 min	142.6 min	58.7 hr	400	6	7
$\mathcal{P}_6$	43.8 min	133.7 min	40.5 hr	146	4	3

TABLE 11. A solution obtained with use of $\mathcal{P}_6$ (Scenario 30).

$\mathcal{P}_i$	TFD	TAD_2	TPD_2	$\#D_2pax$	#Dtrains	$\#D_2sectr$
$\mathcal{P}_1$	34.8 min	108.9 min	24.45 hr	135	2	4
$\mathcal{P}_6$	43.6 min	117.4 min	26.17 hr	88	2	3

TABLE 12. A solution obtained with use of $\mathcal{P}_4$ (Scenario 37).

$\mathcal{P}_i$	TFD	TAD_2	TPD_2	$\#D_2pax$	#Dtrains	$\#D_2sectr$
$\mathcal{P}_1$	59.5 min	221.35 min	72.77 hr	377	8	10
$\mathcal{P}_4$	64.9 min	208.33 min	68.69 hr	275	7	6

comparatively smaller portion of the infrastructure (see Figure 7(b)). As a result, significant improvement is seen with respect to TPD_2 and $\#D_2pax$ (see Table 10).

For disturbance scenario 30, the heuristic algorithm returns an optimal solution with respect to TFD. Using $\mathcal{P}_6$, an additional rescheduling solution with desirable properties is obtained. In this solution, though the TFD and TAD_2 are increased by 8.8 and 8.5 minutes respectively, 47 fewer passengers are delayed (see Table 11).

In Scenario 37, employing the pruning criterion $\mathcal{P}_6$ did not run to completion even within 15 minutes. However, employing $\mathcal{P}_4$ gives an additional solution which slightly increases the value of TFD for an improvement in all other metrics. With an increase in TFD by 5.4 min, this new rescheduling solution is significantly better from a passenger perspective (see Table 12).

VIII. CONCLUSION

This study investigated how a parallel heuristic search algorithm can be used to better solve a real-time railway rescheduling problem while considering multiple perspectives. In this study, solving from a single perspective corresponded to the use of pruning criterion $\mathcal{P}_1$. Solving the problem while considering multiple perspectives meant relaxing the pruning by using multiple metrics.

The conclusions with respect to Proposition 1 are as follows. For the input disturbance scenarios, when pruning is relaxed from criterion $\mathcal{P}_1$ to $\mathcal{P}_6$, the average increase in the number of solution branches explored by the sequential algorithm is 1044%. The number of explored solution branches can increase by as much as 10037% (scenario 24, Table 4). Thus, the algorithm searches a larger number of tree branches, which were otherwise pruned off when criterion $\mathcal{P}_1$ was employed. The analysis presented in Section VII shows that the obtained solution set (when using $\mathcal{P}_6$) often contains several additional desirable solutions, particularly from a passenger perspective.

The conclusions with respect to Proposition 2 are as follows. For the input disturbance scenarios, the speedup attained by parallel multi-objective train rescheduling is not always greater than that attained by parallel single-objective train rescheduling. However, for time-consuming disturbance scenarios, multi-objective rescheduling using the parallel search algorithm led to significant speedups; even greater than the speedups attained by parallel single-objective rescheduling. When pruning based on multiple metrics, larger portions of the search tree are explored, leading to longer execution times, thus making the use of parallel computing very relevant.

Criterion $\mathcal{P}_4$ considers four pruning metrics: total final delay, total accumulated delay, total passenger delay and number of delayed passengers. Criterion $\mathcal{P}_5$ relaxes the pruning by additionally considering the number of delayed trains. Criterion $\mathcal{P}_6$ further relaxes the pruning by also including another metric: number of trains with secondary delays. For few disturbance scenarios, e.g., scenario 24, the use of $\mathcal{P}_5$ and $\mathcal{P}_6$ resulted in no additional solutions in the obtained solution set, compared to the use of $\mathcal{P}_4$. For few scenarios, e.g., scenarios 32, 37, the algorithms (sequential and parallel) did not reach completion even within 15 minutes when using $\mathcal{P}_5$ and $\mathcal{P}_6$. The reason is that a significantly larger portion of the search tree had to be explored when pruning criterion is relaxed to $\mathcal{P}_5$ or $\mathcal{P}_6$. Results indicate that using criterion $\mathcal{P}_4$ is, for several disturbance scenarios, sufficient to obtain a good set of solutions. The obtained solution set often contained additional desirable solutions, e.g., compared to the use of $\mathcal{P}_1$, particularly from a passenger perspective.

Based on the results, we conclude that in the context of train rescheduling and solution space navigation, a parallel tree search algorithm which (i) prunes based on multiple metrics, and (ii) maintains a set of upper bounds, can be beneficial in the following ways. It can improve the quality of the obtained rescheduling solutions and give better speedups with respect to the sequential algorithm.

ACKNOWLEDGMENTS

The research presented in this article has been funded by the following research projects: (i) The TRANS-FORM project, funded by grants (Dnr 942-2015-2034) from the municipality of Karlshamn as well as the Swedish Research Council (FORMAS) via ERA-NET. This project has also received support from Trafikverket, Blekingetrafiken and NetPort. (ii) The FR8Rail II project (grant no: 826206).

REFERENCES

[1] N. Leng and U. Weidmann, "Discussions of the reschedule process of passengers, train operators and infrastructure managers in railway disruptions," *Transp. Res. Procedia*, vol. 27, pp. 538–544, 2017.

[2] S. Binder, Y. Maknoon, and M. Bierlaire, "The multi-objective railway timetable rescheduling problem," *Transp. Res. Part C: Emerg. Technol.*, vol. 78, pp. 78–94, 2017.

[3] J. T. Krasemann, "Computational decision-support for railway traffic management and associated configuration challenges: An experimental study," *J. Rail Transport Planning Manage.*, vol. 5, no. 3, pp. 95–109, 2015.
[4] J. T. Krasemann, "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances," *Transp. Res. Part C: Emerg. Technol.*, vol. 20, no. 1, pp. 62–78, 2012.
[5] S. P. Josyula, J. T. Krasemann, and L. Lundberg, "A parallel algorithm for train rescheduling," *Transp. Res. Part C: Emerg. Technol.*, vol. 95, pp. 545–569, 2018.
[6] S. P. Josyula, J. T. Krasemann, and L. Lundberg, "Exploring the potential of GPU computing in train rescheduling," in *Proc. 8th Int. Conf. Railway Operations Modelling Anal.*, pp. 471–490, 2019.
[7] S. P. Josyula and J. T. Krasemann, "Passenger-oriented railway traffic rescheduling: A review of alternative strategies utilizing passenger flow data," in *Proc. 7th Int. Conf. Railway Operations Modelling Anal.*, 2017.
[8] W. Zhang, "Parallel multi-objective branch and bound," Ph.D. thesis, Tech. Univ. Denmark, 2008.
[9] S. Sudeng and N. Wattanapongsakorn, "Post pareto-optimal pruning algorithm for multiple objective optimization using specific extended angle dominance," *Eng. Appl. Artif. Intell.*, vol. 38, pp. 221–236, 2015.
[10] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.
[11] J. Branke, J. Branke, K. Deb, K. Miettinen, and R. Slowiński, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, vol. 5252. Berlin, Germany: Springer, 2008.
[12] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: Fundamentals and evolutionary methods," *Natural Comput.*, vol. 17, pp. 585–609, 2018.
[13] M. Obara, T. Kashiyama, and Y. Sekimoto, "Deep reinforcement learning approach for train rescheduling utilizing graph theory," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 4525–4533.
[14] M. Shakibayifar, A. Sheikholeslami, and A. Jamili, "A multi-objective decision support system for real-time train rescheduling," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 3, pp. 94–109, Fall 2018.
[15] S. Zhan, L. G. Kroon, J. Zhao, and Q. Peng, "A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage," *Transp. Res. Part E: Logistics Transp. Rev.*, vol. 95, pp. 32–61, 2016.
[16] L. P. Veelenturf, M. P. Kidd, V. Cacchiani, L. G. Kroon, and P. Toth, "A railway timetable rescheduling approach for handling large-scale disruptions," *Transp. Sci.*, vol. 50, no. 3, pp. 841–862, 2016.
[17] M. Ehrgott, "Multiobjective optimization," *AI Mag.*, vol. 29, pp. 47–57, Dec. 2008.
[18] F. Sourd and O. Spanjaard, "A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem," *INFORMS J. Comput.*, vol. 20, no. 3, pp. 472–484, 2008.
[19] A. Przybylski and X. Gandibleux, "Multi-objective branch and bound," *Eur. J. Operational Res.*, vol. 260, no. 3, pp. 856–872, 2017.
[20] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.
[21] Q. Wu, M. Spiryagin, C. Cole, and T. McSweeney, "Parallel computing in railway research," *Int. J. Rail Transp.*, vol. 8, pp. 111–134, 2020.
[22] A. Bettinelli, A. Santini, and D. Vigo, "A real-time conflict solution algorithm for the train rescheduling problem," *Transp. Res. Part B: Methodol.*, vol. 106, pp. 237–265, 2017.
[23] K. Nitisiri, M. Gen, and H. Ohwada, "A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling," *Comput. Ind. Eng.*, vol. 130, pp. 381–394, 2019.
[24] The flatland challenge, 2019. [Online]. Available: https://www.aicrowd.com/challenges/flatland-challenge
[25] H. Khadilkar, "A scalable reinforcement learning algorithm for scheduling railway lines," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 727–736, Feb. 2019.
[26] C. C. McGeoch, *A Guide to Experimental Algorithmics*. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[27] J. Törnquist and J. A. Persson, "N-tracked railway traffic re-scheduling during disturbances," *Transp. Res. Part B: Methodol.*, vol. 41, no. 3, pp. 342–362, 2007.

SAI PRASHANTH JOSYULA is currently working toward the PhD degree in computer science at the Blekinge Institute of Technology, Sweden. His research interests include parallel algorithms, decision-support for train rescheduling, machine learning, etc.

JOHANNA TÖRNQUIST KRASEMANN is an associate professor of computer science at the Blekinge Institute of Technology, Sweden. Her research interests include optimization, algorithms, decision support systems, railway traffic management, etc.

LARS LUNDBERG is a professor of computer science at the Blekinge Institute of Technology, Sweden. His research interests include parallel algorithms, virtual machines, cloud computing, etc.