Transportation Research Part C 95 (2018) 545–569

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Overview Paper

A parallel algorithm for train rescheduling

Sai Prashanth Josyula*, Johanna Törnquist Krasemann, Lars Lundberg

Department of Computer Science, Blekinge Institute of Technology, Valhallavägen 1, 371 41 Karlskrona, Sweden

ARTICLE INFO

Keywords:
Railway traffic
Rescheduling
Parallel depth-first search
Optimization

ABSTRACT

One of the crucial factors in achieving a high punctuality in railway traffic systems, is the ability to effectively reschedule the trains when disturbances occur. The railway traffic rescheduling problem is a complex task to solve both from a practical and a computational perspective. Problems of practically relevant sizes have typically a very large search space, making them time-consuming to solve even for state-of-the-art optimization solvers. Though competitive algorithmic approaches are a widespread topic of research, not much research has been done to explore the opportunities and challenges in parallelizing them. This paper presents a parallel algorithm to efficiently solve the real-time railway rescheduling problem on a multi-core parallel architecture. We devised (1) an effective way to represent the solution space as a binary tree and (2) a novel sequential heuristic algorithm based on a depth-first search (DFS) strategy that quickly traverses the tree. Based on that, we designed a parallel algorithm for a multi-core architecture, which proved to be 10.5 times faster than the sequential algorithm even when run on a single processing core. When executed on a parallel machine with 8 cores, the speed further increased by a factor of 4.68 and every disturbance scenario in the considered case study was solved within 6 s. We conclude that for the problem under consideration, though a sequential DFS approach is fast in several disturbance scenarios, it is notably slower in many other disturbance scenarios. The parallel DFS approach that combines a DFS with simultaneous breadth-wise tree exploration, while being much faster on an average, is also consistently fast across all scenarios.

1. Introduction

Decision-making is the process of identifying, assessing and making appropriate decisions to solve a problem. Scheduling is a decision-making process that involves making choices regarding allocation of available resources to tasks over a given time period with a goal to optimize one or more objectives (Pinedo, 2016). Scheduling is a frequently employed crucial operation in several organizations and sectors e.g., manufacturing industries and the railway transport sector.

In railway traffic network management, the ability to efficiently schedule the trains and the network maintenance, influences the punctuality of trains and Quality of Service (QoS) significantly. The importance is reflected in the goal set by the Swedish railway industry stating that by year 2020, 95 % of all trains should arrive at the latest within five minutes of the initially planned arrival time (Trafikverket, 2015). Similar goals have been set by the railway industries in Australia (NSW-Transport, 2016), Netherlands (Nederlandse-Spoorwegen, 2016) and several countries across the world, thus emphasizing the importance of train punctuality and QoS. The punctuality of trains is primarily affected by (1) the occurrence of disturbances, (2) the robustness of the train schedules (i.e., the timetables) and the associated ability to recover from delays, as well as (3) the ability to effectively reschedule trains when

* Corresponding author.
E-mail addresses: sai.prashanth.josyula@bth.se (S.P. Josyula), johanna.tornquist.krasemann@bth.se (J. Törnquist Krasemann), lars.lundberg@bth.se (L. Lundberg).

https://doi.org/10.1016/j.trc.2018.07.003
Received 28 November 2017; Received in revised form 27 June 2018; Accepted 4 July 2018

disturbances occur, so that consequences, e.g., delays, are minimized.

In this paper we focus on the latter, and present an algorithm for efficient rescheduling of railway traffic during disturbances. The purpose of the algorithm is to compute, in a short time, a relevant set of alternative revised schedules to support the train traffic dispatchers in the real-time decision-making. In order to benefit from the advances in computer hardware, and with an aim of generating revised schedules of good quality faster, we design and implement corresponding parallel algorithms for the initially designed algorithm.

The paper is organized as follows: In the next section, we describe the rescheduling problem in more detail and the scope of this study. In Section 3, we present an overview of related research work and a brief discussion of the main research challenges addressed in this paper, along with the expected research contributions. In Section 4, we present the basic terminology used and the design of the algorithmic approach, along with the rationale for crucial design choices and the types of rescheduling decisions that are applied by the algorithm. We conclude the section with a description and discussion of the parallel algorithm. In Section 5, we describe the experimental platform, the chosen case study, and the key aspects that are considered for performance evaluation. In Section 6, we present, analyze and discuss the results. We present conclusions and suggested future work in Section 7.

2. Problem description

In the railway sector, day-to-day 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 disturbance in a railway network is an unexpected event that renders the originally planned timetable infeasible by introducing 'conflicts'. A conflict is considered to be a situation that arises when two trains require an infrastructure resource during overlapping time periods in a way such that one or more system constraints are violated.

Disturbances are triggered by incidents such as over-crowded platform(s) that possibly lead to unexpectedly long boarding times and minor delays, or larger incidents such as power shortages, train malfunctions, signalling system failures that cause more significant delays. Railway timetables are planned with appropriate time margins in order to recover from minor delays. Hence, in case of a minor disturbance, 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 original timetable and it becomes operationally infeasible. The resolution of these conflicts to obtain a feasible timetable during operations, constitutes real-time railway traffic rescheduling.

In order to resolve a conflict, the following three tactics are frequently employed: (1) Retiming, i.e., allocating new arrival and departures times to one or more trains, (2) local rerouting, i.e., allocating alternative tracks to one or more trains, (3) reordering, i.e., prioritizing a train over another. Apart from these tactics, conflicts can also be handled by: (4) Globally rerouting the trains, or (5) partially/fully cancelling the affected train services. The algorithmic approach presented in this paper applies only the first three mentioned rescheduling actions.

During a disturbance, rescheduling the railway traffic is typically handled manually by train dispatchers who have very limited access to decision support systems (Törnquist Krasemann, 2012; Larsen et al., 2014). The time available for analysing alternative decisions is often very limited. Under these circumstances, a safe rescheduling strategy often employed by train dispatchers is to reduce the delay of important trains by prioritizing them over other trains (Törnquist Krasemann, 2012). 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.

3. Related work

In this section, we present an overview of the work that is most relevant to our research objectives. The various problem formulations, models and solution approaches employed for real-time railway (re) scheduling have been surveyed time and again by researchers (Törnquist, 2006; Cacchiani et al., 2014; Fang et al., 2015). In recent work, Fang et al. (2015) present a comprehensive survey of various types of modelling and solution approaches for the railway rescheduling problem. According to their survey, the most frequently used models for the rescheduling of railway traffic networks are mixed integer linear programming (MILP) models, alternative graph (Mascis and Pacciarelli, 2002) models and integer programming (IP) models, in the mentioned order. The survey by Fang et al. (2015) also reveals that heuristic approaches are most frequently employed by researchers to solve real-time railway rescheduling problems.

Real-time railway rescheduling can be considered as a combinatorial optimization problem. In combinatorial optimization, Graphics Processing Unit (GPU) computing has been successfully used by meta-heuristic algorithms (Bożejko et al., 2010, 2012; Luong et al., 2013) as well as exact algorithms (Melab et al., 2012; Dabah et al., 2016) to achieve significant speedups. Bożejko et al. (2010) report about achievements of significant speedups ($2\times$–$55\times$) for various benchmark instances of the flexible job shop problem, by parallelizing their algorithm on GPUs.

The branch and bound algorithmic approach to the flow shop problem has also been parallelized on the GPU in recent works (Melab et al., 2012; Dabah et al., 2016). Melab et al. (2012) use the computing power of the GPU for the calculation of lower bounds (rather than using it for the parallel exploration of the search tree). Their approach is well-motivated as the explored search tree is highly irregular, thus making the tree exploration not well-suitable for parallelization on the GPUs. The algorithm uses a large pool of threads on the GPU to compute the lower bounds while the CPU performs elimination, selection and branching operations. The