Quantum computing approach to railway dispatching and conflict management optimization on single-track railway lines

Krzysztof Domino\textsuperscript{1}, Mátyás Koniorczyk\textsuperscript{2}, Krzysztof Krawiec\textsuperscript{3}, Konrad Jałowiecki\textsuperscript{4}, and Bartłomiej Gardas\textsuperscript{4}

\textsuperscript{1}Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland
\textsuperscript{2}Wigner Research Centre, Budapest, Hungary
\textsuperscript{3}Silesian University of Technology, Faculty of Transport and Aviation Engineering, Gliwice, Poland
\textsuperscript{4}Institute of Physics, University of Silesia, Katowice, Poland

October 16, 2020

Abstract

We consider a railway dispatching problem: delay and conflict management on a single-track railway line. We examine the issue of train dispatching consequences caused by the arrival of an already delayed train to the segment being considered. This is a computationally hard problem and its solution is needed in a very short time in practice. We introduce a quadratic unconstrained binary optimization (QUBO) model of the problem that is suitable for solving on quantum annealer devices: an emerging alternative computational hardware technology. These offer a scalability of computationally hard problems better than that of classical computers. We provide a proof-of-principle demonstration of applicability through solving selected real-life problems from the Polish railway network. We carry out the actual calculation on the D-Wave 2000Q machine. We also include solutions of our model using classical algorithms for solving QUBO, including those based on tensor networks. These are of potential practical relevance in case of smaller instances and serve as a comparison for understanding the behavior of the actual quantum device.

Keywords

Railway dispatching problem, train delay management, railway conflict management, Quadratic Unconstrained Binary Optimization (QUBO), quantum annealing, adiabatic quantum computing, quantum-inspired algorithms, tensor network simulations.

1 Introduction

Railway operations involve a broad range of scheduling activities, ranging from operational train dispatching to provisional timetable planning in case of disturbances. Many of these require solving computationally expensive and overall difficult combinatorial problems. The consequences of incorrect dispatching decisions may be serious in term of various resources (time costs, passengers’ satisfaction, financial loses, etc.).

For this importance of solving hard problems, new computational devices developed for the purpose, like those exploiting real or simulated quantum systems, are potentially useful in this branch of operations and transportation research. The problems in which the best possible real-time solutions are needed are especially interesting because of the limited time available for the calculation. However, new devices require different mathematical modeling frameworks. In particular, adiabatic quantum computers, such as D-Wave annealers, promise to solve Quadratic Unconstrained Binary Optimization (QUBO) problems with a scaling better than of classical computers. These quantum computers are expected to provide admissible solutions (encoded in the ground state of the Ising spin-glass
model) in manageable time also for larger problems (currently, up to $N = 2048$ variables on a sparse connectivity graph called Chimera [1]).

A priori, each spin configuration can be a potential ground state. The number of possible combinations that is “searched” by these devices is so enormous (i.e., $2^{2048} \approx 10^{615}$) that it can barely be captured by our imagination [2]. To put things in perspective, it is worth mentioning that searching through all configurations classically in a reasonable time is only possible for small system sizes (that is, using present-day state of the art technology). Even then, incorporating massive parallelization techniques, it takes roughly two weeks to find the low-energy spectrum of the Ising Hamiltonian with only 50 spins [3]. In comparison, annealers process information on a time scale of 5µs–2ms, roughly independent on the number of qubits [4]. This highlights the tremendous potential of these machines that could be harnessed to perform extremely fast computations.

Apart from the applicability of actual quantum devices, there are efficient variational algorithms that have been developed in the last few decades, primarily to tackle problems in statistical and solid-state physics. Given a QUBO model, these approaches can also be applied to successfully tackle the above-mentioned problems, as a potential alternative for solving using the classical algorithms.

The goal of our work is to address the methodology discussed above and implement a proof-of-principle application of QUBO in a real-life situation. We opt for the problem of delay and conflict management on a railway network. The problem is NP-hard and the real-life solution needs to be available as soon as possible, hence we find it ideal for the demonstrative application of the algorithm.

This paper is organized as follows. In Section 2 we present the literature review focused on solving railway dispatching problems 2.1 and the quantum annealing and its emulation 2.2. In Section 3 we introduce the model of trains delays management, concerning the railway line 3.1, the delay management and the objective function 3.2 and the dispatching conditions 3.3. In Section 4 we formulate above-mentioned model in the form of the Quadratic Unbound Binary Optimization representation, by introducing the 0-1 programming representation 4.1, penalty methods 4.2, and demonstrating it on the the simple illustrative example 4.3. In Section 5 we present numerical studies: we describe the railway lines in 5.1, we solve dispatching problems therein by conventional heuristics in 5.2 and by quantum and quantum inspired approach in 5.3. In Section 6 we provide discussion and conclusions. Finally, in Appendix A we present train diagrams of particular solutions.

2 Literature review

Here we revise the scientific context of our work. This section is divided into two parts. In the first one we clarify the terminology used throughout the paper, describe the problem, and categorize it according to the standard notation of scheduling theory. We present some of the existing approaches based on the literature and position our problem in this scientific context. In the second part we summarize the theoretical background necessary to understand the quantum annealing method, including Ising-based solvers, quantum annealing, and classical algorithms for solving Ising problems.

2.1 Railway dispatching problem on the single-track lines

Railway dispatching problem management is quite a complex area of transportation research. Here we focus on the delay management on the single-track lines. This problem concerns the operative modifications of train paths upon disturbances in the railway traffic. Inappropriate decisions may cause the deterioration of the dispatching situation, namely delay propagation and its consequences. Below we give the details of the problem and revise a part of the literature on the topic. While we focus on single-track railway lines, some of the considerations may also be applied for multi-track railways too.

2.1.1 Problem description

The situation to be handled is as follows. There is a part of a railway network in which the traffic is affected by some disturbance, and thus some trains cannot run according to the original timetable. We are not concerned with the actual reason. Hence, a new, feasible timetable should be designed promptly, reducing unwanted consequences of the delay as much as possible.

In more detail, we are given a part of a railway network (referred to as ‘the network’). The network is divided into block sections. This term originates in the railway signaling terminology and in general it means a section of the railway line between one signal box and another. In our problem, a block section is understood as a railway network section that can only be occupied by one train at a time. Note that sometimes, a term track section is used instead. However, since track section is also used to describe different notions, we shall avoid its use to avoid confusion. (Sometimes, a term track section is used in our meaning of block section albeit it also has different meanings.)

We focus on single-track railway lines. These include passing sidings (referred to as ‘sidings’): parallel tracks, typically at stations, where trains heading opposite directions can meet and pass (M-P), while the trains heading the same directions can meet and overtake (M-O). Later we shall point out that there are multiple ways of including
sidings to the model. The implications of an adverse decision can be serious in terms of time on these lines because of the time needed for a train to reach subsequent sidings.

Trains run according to a timetable. We assume that an initial timetable is conflict free and meets all the feasibility criteria. There can be different variants of these criteria [5, 6], depending on the actual railway network including e.g., technological requirements such as speed limits, dwell times, and other signaling-imposed requirements, rolling stock circulation criteria, passenger demands for trains to meet, etc. By a conflict we understand the inadmissible situation whereby at least two trains are supposed to occupy the same block section.

The railway delay management problem can be viewed from various perspectives, including that of a passenger, the infrastructure manager, or transport operation company [5, 6, 7]. Here we look at this problem from the traffic management perspective of the infrastructure manager who is to make the ultimate decision about the modifications, and is in the position to prioritize the requirements.

In our problem, we assume that – for whatever reason – a delay occurs. Hence, at the beginning of the analysis, some of the trains’ locations are different from the scheduled ones. The goal is to redesign the timetable to get a feasible one, avoiding conflicts, and minimizing delays. There are two types of delays, though. A primary delay is unavoidable. It is calculated for each train based on the temporary assumption that no other trains are present on the network, thus no conflict may occur and the whole infrastructure can serve that single train. The secondary delay is the delay beyond the primary delay, typically caused by conflicts, but also influenced e.g. by priorities of trains. A cumulative version (e.g. maximum or a weighted sum) of the secondary delays is to be minimized. We remark here that in railway technology, the terms ‘primary delay’ and ‘secondary delay’ are often used in another sense: a primary delay has its reasons in external circumstances (e.g. heavy weather) or unplanned events (e.g. breakdown of a locomotive), while secondary delays in this sense are consequences of primary delays of some trains.

We shall, however, use the terminology common in the literature of our problem. We also remark that there are many other practically relevant options for an objective [8] including e.g. the total passenger delay or the cost of operations, some of which are also in line with our approach.

The mathematical treatment of railway delay and conflict management leads to NP-hard problems; certain simple variants are NP-complete [9]. It is broadly accepted that these problems are equivalent to job-shop models with blocking constraints [10], given release and due dates of the jobs and, depending on the requirements of the model, additional constraints such as recirculation or no-wait constraints. The correspondence between the metaphors is the following. Trains are the jobs, block sections are the machines. Concerning objective functions, (weighted) total tardiness or make-span are typically addressed, which are the (weighted) sum of secondary delays or the minimum of the largest secondary delay in the railway context. So with the standard notation of scheduling theory [11], our problem falls into the class $J_m|r_i,d_i,block|\sum_j w_j T_j$.

2.1.2 Existing approaches

In our summary, we focus on the concepts and methods we consider as closely related to our treatment. A more comprehensive review of the tremendous literature of optimization methods applicable to the railway management problems can be found in numerous related publications, notably [6, 8, 12, 13, 14, 15, 16].

On the single-track line, one may reschedule only the trains by allocating new arrival and departure times, change tracks and platforms, and also re-order the trains by adjusting the needed meet-pass plans [17, 6, 8]. An important issue related to the single-track lines is handling the sidings (stations) in the model. As pointed out in [18], there are three approaches to tackle this:

- **Parallel Machine Approach** assumes that each track (in our notation station block) within the siding corresponds to a separate machine in the job shop, thereby losing the possibility of flexible routing i.e. changing track orders at a station afterward.
- **Machine unit approach**, treating parallel tracks (section blocks) as additional units of the same machine
- **Buffer approach**, handling sidings at the same location as buffers without internal structure, therefore not warranting the feasibility of track occupation planning at a station.

As of the nature of the decision variables, two substantial classes of models may be identified:

- **Order and precedence variables** prescribe the order in which a machine processes jobs, that is, the order of trains passing a given block section in the railway dispatching problem on the single-track lines.
- **Discrete time units**, in which the decision variables belong to discretized time instants: the binary variables describe whether the event happens at a time given, or not.

These two approaches lead us to the different model structures which we find hard to compare. The discrete time units approach appears to be more suitable for a possible QUBO formulation, but it leads to a large number of decision variables, and thus a worse scaling. On the other hand, order and precedence variables can lead to a representation of the problem with alternative graphs [19, 20] which is an intuitive picture. Their solution leads to mixed-integer programs that can possibly solved with iterative methods (such as branch-and-bound), which makes them unsuitable for a reformulation to QUBO. Time indexed variables on the other hand results in the pure integer or binary problems, which the exact solution can be beyond the possibilities of classical algorithms. However, they can be transformed into QUBOs [21], so we follow the latter approach.
Returning to [18], the considered problem with the parallel machines approach, machine unit approach, and order and precedence variables is in all the cases – denoted as follows $J_m[r, d, block, recr] \sum T_j$ when using the standard notation of scheduling theory set out in [11]. We will adopt slightly different constraints and objectives. Namely, $J_m[r, d, block] \sum w_j T_j$. Moreover, the cited models explicitly ignore deadlocks, that is, when two counter-heading trains meet at the connecting point of two adjacent block sections. In this work, deadlocks are solved by track interchange to simplify the solution. As we intend to deal with real-life railway systems, in which such an infeasibility is not acceptable, we will not assume such a simplification. Also, as we opt for the discrete-time unit and time-indexed variables. Thus, for this reason, let us turn the attention to this scope in more depth.

In [22], Zhou, and Zhong have considered the problem of timetabling on a single-track line. The starting times of trains and their stops are given, and a feasible schedule is to be designed to minimize the total running time of (typically passenger) trains. While their problem, notably the objective function and the inputs are different, the constraints are similar to those of ours. They also deal with conflicts, dwell times, and minimum headway times for entering a segment of the railway line. They handle the problem with reference to resource-constrained project scheduling. Their decision variables are the discretized entry and leave times of the trains at the track segments, binary precedence variables describing the order of the trains passing a track segment, and time-indexed binary variables describing the occupancy of a segment by a given train at the given time. They introduce a branch-and-bound procedure with an efficiently calculable conflict-based bound in the bounding step to supplement the commonly used Lagrangian approach. They demonstrate its applicability for scheduling of up to 30 passenger trains for a 24 hour periodic planning horizon on a line with 18 stations in China.

Harrod [23] has proposed a discrete-time railway dispatching model, with a focus on a conflict management. A train traffic flow is modeled as a directed hypergraph, where hyperarcs represent train moves with various speeds. This may be confined to an integer programming model with time, train, hypergraph-related variables, and a complex objective function covering multiple aspects. The model is demonstrated on an imaginary single-track line with long passing sidings at even numbered block sections. It is of 19 blocks of length. An intensive flow of trains at moderate speeds is examined. The model instances are solved by CPLEX in order of 1000 seconds of computation time. As a practical application, a segment of a busy North American mainline is used, on which it produced practically useful results. Bigger instances were also experimented with, leading to a conclusion that the approach is promising but needs more specialized technology than a standard MIP solver to be efficient.

Meng and Zhou [24] describe a simultaneous train rerouting and rescheduling model based on network cumulative flow variables. Their model also employs discrete-time-indexed variables. They implement a Lagrangian relaxation solution algorithm and make detailed experiments showing that their approach performs promisingly on a general n-track railway network. The introduction of this reference tabulates the numerous timetabling and dispatching algorithms.

The brief survey of the extensive bibliography confirms that the problems of railway dispatching and conflict management is indeed a good candidate for a demonstration of a new computational technology capable of solving hard problems. Only a very few models have actually been put into practice. The size and complexity of realistic daily dispatching problems makes them almost never solvable using current computational technology.

Quantum and quantum-inspired algorithms have already been used for the railway system’s investigation, but rather for the rolling stock rostering [25]. Our approach is based on the job shop problem, and is inspired by the general approach proposed in [21].

2.2 Quantum annealing and related methods

Let us now turn our attention to the main tools used in the present investigation: quantum annealing techniques, the new computational paradigm, which under additional assumptions [26], is equivalent to general purpose quantum computation. Thus, this emerging technology promises to tackle complicated (NP-hard in fact [27]), discrete optimization problems by encoding them in the ground state of a physical system. Namely, the Ising spin-glass model [28]. Such a system is then allowed to reach its ground state "naturally" via an adiabatic-like process [4].

2.2.1 Ising-based solvers

The Ising model, introduced originally for the microscopic explanation of magnetism, has been in the heart of research interest ever since. It deals with a set of variables $s_i \in \{+1, -1\}$ (corresponding to direction of microscopic magnetic momenta associated to spins originally). The configuration of $N$ such variables is thus described by a vector $s \in \{+1, -1\}^N$. The model then assigns the energy to a particular configuration:

$$E(s) = \sum_{(i,j) \in E} J_{i,j} s_i s_j + \sum_{i \in V} h_i s_i,$$  \hspace{1cm} (1)

where $(V, E)$ is a graph whose vertices $V$ represent the spins, the edges $E$ define which spins interact, $J_{i,j}$ is the strength of this interaction, and $h_i$ is an external magnetic field at spin $i$. While the early studies of the model had dealt with configurations in which the spins were arranged in an one-dimensional chain so that the coupling $J$
Figure 1: D-Wave processor specification. Left: An example of the chimera topology, here comprising of a $4 \times 4$ ($C_4$) grid consisting of clusters (units cells) of 8 qubits each. The total number of variables (vertices) for this graph is $N = 4 \cdot 4 \cdot 8 = 128$. Graph’s edges indicate possible interactions between qubits. The maximum number of qubits is $N_{\text{max}} = 2048$ for the chimera $C_{16}$ topology whereas the number of all connections between them is limited to only $600 \ll N_{\text{max}}^2$. Right: A typical annealing schedule controlling evolution of a quantum processor, where $T$ denotes time to complete one annealing cycle (the annealing time). It ranges from microsecond ($\sim 2\mu s$) to milliseconds ($\sim 2000\mu s$). The parameters $g$ and $\Delta$ are used in (7).

was not zero for nearest neighbors only, the model was generalized many ways including the most general setting of an arbitrary $(V,E)$ graph, that is, the possibly nonzero couplings for any $i,j$ pair. Such a system is referred to as a spin glass in physics, and it is the very physical model which is interesting from operations research point of view, for the system it describes is computational resource for optimization. The idea originates from the fact that in physics the minimum energy configuration determines many properties of the material, hence, a lot of effort has been made to find it. In addition, the minimum energy configuration can be realized in actual physical systems, thus a special hardware – a so-called (quantum) annealer – can also be constructed.

In mathematical programming it is often more convenient to deal with 0-1 variables. By introducing new decision variables $x \in \{0, 1\}$ so that

$$x_i = \frac{s_i + 1}{2}$$

and the matrix

$$Q_{i,i} = 2 \left(h_i - \sum_{j=1}^{n} J_{i,j}\right)$$

$$Q_{i,j} = 4J_{i,j},$$

the minimization of the energy function is equivalent to solving a quadratic unconstrained binary optimization problem (QUBO for short):

$$\min. \quad y = x^T Q x,$$

$$\text{s.t.} \quad x \in \{0, 1\}^N.$$ 

(4)

Therefore, minimizing the Ising-objective in (1) is equivalent to solving a QUBO. Moreover, the matrix $Q$ can always be chosen to be symmetric, as $Q = (Q + Q^T)/2$ defines the same objective. Solvers based on Ising spin-glasses are actual devices (or specialized algorithms simulating them or calculating their relevant properties) that can handle models of this form only. The technology offers the possibility of efficiently tackling computationally hard problems (when formulated as a QUBO which is possible for all linear or quadratic 0-1 programs [ ]). It does have limitations with respect to size and accuracy as it will be illustrated in the present case study, but is likely that it will continue to improve.

Simultaneously, with the rapid development of quantum annealing technology, probabilistic classical accelerators have been under massive development. In recent years, we have witnessed significant progress in the field of programmable gate array optimization solvers (digital annealers [29]), optical Ising simulators [30], coherent Ising machines [31], stochastic cellular automata [32], and in general, those based on memristor electronics [33]. It is therefore vital to develop modeling strategies to compile operational problems to such models and create novel techniques to analyze the obtained results further. This should progress similar to that which went on when powerful solvers for linear programs first started appearing: modeling strategies for linear programs as well as the sensitivity analysis was developed ahead of hardware being created.

2.2.2 Quantum annealing

An essential step in finding the minimum of an optimization problem [encoded in (1)] efficiently is to map it to its quantum version. The mapping assigns a two-dimensional complex vector space to each spin and a complete spin
configuration becomes an element of the direct (tensor) products of these spaces. An orthonormal basis is assigned to the \(-1\) and \(+1\) values of variables; thus the product of these vectors will be an ONB (Orthonormal Basis - termed as the ‘computational basis’) in the whole \(\mathbb{C}^{2N}\). The vectors with unit Euclidean norms are referred to as ‘states’ of the system; they encode the physical configurations. The fact that the state can be an arbitrary vector, not only an element of the computational basis means that the quantum annealer can simultaneously process multiple configurations: inherent parallelism is manifested.

As of the objective function, the spin variables are replaced by their quantum counterpart: Hermitian matrices acting on the given spin’s \(\mathbb{C}^{2}\) tensor subspace:

\[
s_i \mapsto \hat{\sigma}^i = \text{diag}(1, -1)
\]

where the matrix represents the respective operator in the computational basis. The product of spins is meant as the direct (tensor) product of the respective operators. Thus the objective function (1) turns into a Hermitian operator, referred to as the problem’s Hamiltonian:

\[
\mathcal{H}_p := E(\hat{\sigma}^a) = \sum_{\langle i,j \rangle \in \mathcal{E}} J_{ij} \hat{\sigma}^i_+ \hat{\sigma}^j_+ + \sum_{i \in \mathcal{V}} h_i \hat{\sigma}^i_z,
\]

whose lowest-energy eigenstate is commonly termed as a “ground state.” In the present case, it is an element of the computational basis, and thus it represents also the optimal configuration of the classical problem. Note, the energy of a physical system is related (via eigenvalues) to a Hermitian operator, called its Hamiltonian. While it seems to be a significant complication to deal with \(\mathbb{C}^{2N}\) instead of having \(2^N\) binary vectors it has important benefits; most remarkably that they model actual realistic physical systems and thus they are realized by nature.

The main idea behind quantum annealing is based on the celebrated adiabatic theorem [34]. Broadly speaking, when a quantum system (starting from its ground state) is driven (i.e., its Hamiltonian is adjusted in time) slowly enough so that it has time to adjust to changes it can remain in its ground state during the entire evolution. At the end, a solution to a computational problem can be read out from the final ground state. As an aside note it should be stressed that this scenario does not work in some cases. Take for instance, the second–order phase transition phenomenon [35, 36, 37]. Even a short–lasting lack of adiabaticity will result in the creation of topological defects preventing the system from remaining in its ground state. This effect, which results from the Kibble–Żurek mechanism [38, 39, 40], can be then used to detect departures from adiabaticity.

To be more specific, assume a quantum system can be prepared in the ground state of an initial (“simple”) Hamiltonian \(\mathcal{H}_0\). Then it will slowly evolve to the ground state of the final (“complex”) Hamiltonian \(\mathcal{H}_p\) in (6) that one can harness to encode the solution of an optimization problem [28]. In particular, the dynamics of the current D-Wave 2000Q quantum annealer is governed by the following time-dependent Hamiltonian [4, 41]

\[
\mathcal{H}(t)/\hbar = -g(t)\mathcal{H}_0 - \Delta(t)\mathcal{H}_p', \quad t \in [0,T].
\]

Here the original problem Hamiltonian in (6) has to be transformed to an equivalent one, \(\mathcal{H}_p'\). The reason for this is that the existing hardware can realize only a specific graph configuration, termed as the ‘Chimera graph’, cf. Fig. 1. Thus a mapping of the original problem to a one having typically more spins but compliant with the chimera graph (\(\mathcal{E}, \mathcal{V}\)) is needed. (See Section 4.3 for the simple graphical representation of this chimera embedding).

In fact, many relevant optimization problems are defined on dense graphs. Fortunately, even complete graphs can be embedded onto the chimera graph [42]. There is, however, a substantial overhead that effectively limits the size of the computational graph that can be treated with current quantum annealers [43, 44]. This is, nevertheless, believed to be an engineering issue that will most likely be overcome in the near future [1, 45]. So after the chimera embedding, the Hamiltonian describing the actual physical system is

\[
\mathcal{H}_p' = \sum_{\langle i,j \rangle \in \mathcal{E}} J'_{ij} \hat{\sigma}^i_+ \hat{\sigma}^j_+ + \sum_{i \in \mathcal{V}} h' \hat{\sigma}^i_z \quad \text{and} \quad \mathcal{H}_0 = \sum_i \hat{\sigma}^i_x,
\]

where

\[
\hat{\sigma}^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

and \(\hat{\sigma}^z\) is defined in (5). The annealing time \(T\) varies from microseconds (~2\,\text{ms}) to milliseconds (~2\,\text{ms}) depending on the programmable schedule [46]. As shown in Fig. 1, during the evolution \(g(T)\) varies from \(g(0) \gg 0\) (i.e. all spins point in the \(x\)-direction) to \(g(T) \approx 0\) whereas \(\Delta(t)\) is changed from \(\Delta(0) \approx 0\) to \(\Delta(T) \gg 0\) (i.e. \(\mathcal{H}(T) \approx \mathcal{H}_p'\)). The Pauli operators \(\hat{\sigma}^i_x, \hat{\sigma}^i_z\) describe the spin degrees of freedom in the \(z\)- and \(x\)-direction respectively. Note, the Hamiltonian \(\mathcal{H}_p'\) is classical in a sense that all its terms commute (that is the result of their multiplicity is independent of the order). Thus, as mentioned previously, its eigenstates translate directly to classical variables, \(q_i = \pm 1\) which are introduced to encode discrete optimization problems.

Adiabatic quantum computation may be viewed as an alternative paradigm of computing [47]. Essentially, it is equivalent to the gate model of quantum computation (up to a polynomial penalty) that uses logical gates operating on quantum states to implement quantum algorithms [26]. This is, of course, provided that the specific interactions between quantum bits can be realized experimentally [48].
2.2.3 Classical algorithms for solving Ising problems

An additional benefit of formulating problems in terms of Ising-type models is that all the rather sophisticated methods developed to find ground states of physical systems can be exploited to solve a QUBO on a classical hardware. Notably, variational methods based on finitely correlated states (such as matrix-product states for 1D systems or Projected Entangled Pair States suitable for 2D graphs) had a very extensive development in the last few decades. A quantum-information theoretic insight into density-matrix renormalization group methods (DMRG [49]) – being the most powerful numerical techniques in solid state physics to that point – helped in proving the correctness of DMRG. They also lead to a more general view of the problem [50] resulting in many algorithms that have potential applications in various problems, in particular solving QUBOs by finding the ground state of a quantum spin glass. We have used the algorithms presented in the detailed study in [51] to solve the models derived in the present manuscript.

Both quantum computers and the mentioned classical algorithms have the chance not to provide the energy minimum and the corresponding ground state, but another eigenstate of the problem with an eigenvalue (that is, a value of the objective function) close to the minimum. (In fact it is not at all trivial to reach the ground state; see also [52].) The corresponding states are referred to as ‘excited states’. In the physical computer it is due to the inevitable noise present in the system, whereas in the variational algorithms it is due to their approximating nature. Hence, when using such a solver the interpretation of the results should address this possibility. The other important point when interpreting the results coming from such a solver is that of the degeneracy of the solution: the possibility to have multiple equivalent optima.

In analyzing the above mentioned issue it is very helpful that for up to 50 variables one can calculate the exact ground states and the excited states closest to it using a brute-force search on the spin configurations with GPU-based high performance computers. In the present work we also use such algorithms, in particular those introduced in [53] for benchmarking and evaluating our results for smaller instances. This way we can compare the exact spectrum with the results obtained from the D-Wave quantum hardware and the variational algorithms.

3 Our model

Let us now discuss our model, including the description of the railway line and the dispatching conditions. Tab. 1 provides a comprehensive notation summary.

3.1 Railway line

We assume a railway line \( \mathcal{M} \) to be a set of block sections (see Section 2.1.1). These are either line blocks or station blocks, both referred to also as block sections or just blocks. We denote the set of line blocks as \( \mathcal{L} \), and the set of station blocks as \( \mathcal{S} \). This model also incorporates sidings or double track sections by treating them as station blocks.

Trains can only meet and pass (M-P) or meet and overtake (M-O) at stations. We follow the buffer approach by treating each station as a block that can be occupied up to \( S \) trains at a time, where \( b \) is a number of tracks at the station. The other blocks can only be occupied by one train at a time.

The set of trains is denoted by \( \mathcal{J} \) and is split into the subset of trains traveling in a given direction \( \mathcal{J}^0 \) and the subset of trains going in the opposite direction \( \mathcal{J}^1 \):

\[
\mathcal{J}^0 \cup \mathcal{J}^1 = \mathcal{J} \quad \text{and} \quad \mathcal{J}^0 \cap \mathcal{J}^1 = \emptyset.
\]

Let \( j \in \mathcal{J} \) be a particular train. Its route is a sequence of blocks \( M_j = (m_{j,1}, m_{j,2}, \ldots , m_{j,\text{end}}) \), where \( m_{j,1} \) is the starting block and \( m_{j,\text{end}} \) is the ending block. Each block (from \( M_j \)) is passed by the train \( j \) once and only once (i.e., we do not consider recirculation). Given a train \( j \) and a block \( m_{j,k} \) the preceding block is \( \pi_j(m_{j,k}) = m_{j,k-1} \) while the subsequent block is \( \rho_j(m_{j,k}) = m_{j,k+1} \). We assume that neither \( \rho_j(m_{j,\text{end}}) \) nor \( \pi_j(m_{j,1}) \) belong to the analyzed network segment.

We assume that route can be solely defined by a sequence of the station blocks \( S_j = (s_{j,1}, s_{j,2}, \ldots , s_{j,\text{end}}) \) where M-P and M-O may occur (that is, there are no different alternative routes between stations). Similarly to blocks in general, for a train \( j \) and a station block \( s_{j,k} \) we denote the preceding station block as \( \pi_j(s_{j,k}) = s_{j,k-1} \), and the subsequent station block as \( \rho_j(s_{j,k}) = s_{j,k+1} \). It is convenient to assume that all train paths to be analyzed start and end at stations, hence we have \( s_{j,1} = m_{j,1} \) and \( s_{j,\text{end}} = m_{j,\text{end}} \).

3.2 Delays and the objective function

Ideally, the time \( t_{\text{out}}(j,s) \) the train \( j \) leaves the station block \( s \) should equal to the timetable one \( t_{\text{out}}^{\text{timetable}}(j,s) \). If, however, \( t_{\text{out}}(j,s) > t_{\text{out}}^{\text{timetable}}(j,s) \) we have a delay while leaving the station block \( s \):

\[
d(j,s) = t_{\text{out}}(j,s) - t_{\text{out}}^{\text{timetable}}(j,s).
\]
| symbol | description / explanation |
|--------|---------------------------|
| \( A_{j,s} \) | discretized set of all possible delays of the train \( j \) at the station \( s \) |
| \( \mathcal{H}(t), \mathcal{H}_0, \mathcal{H}_p \) | Hamiltonian depending on the processing time instant, and its time independent components |
| \( t \in [0,T] \) | the time instant of the processing time on the quantum machine |
| \( \hat{\sigma}^x, \hat{\sigma}^z \) | Pauli matrices, here \( x, y, z \) are physical coordinates |
| \( j \in \mathcal{J} \) | trains (jobs) |
| \( \mathcal{F}^0 (\mathcal{F}^1) \) | trains going a given (opposite) direction |
| \( m \in \mathcal{M} \) | blocks (machines) |
| \( s \in \mathcal{S} \) | station blocks |
| \( l \in \mathcal{L} \) | line blocks |
| \( M_j, (S_j) \) | the sequence of blocks (station blocks) in the route of \( j \) |
| \( s_{j,1}, s_{j,k}, s_{j,end} \) | first, \( k \)-th, last station block in the route of \( j \) |
| \( m_{j,1}, m_{j,k}, m_{j,end} \) | first, \( k \)-th, last block in the route of \( j \) |
| \( S_j = (s_{j,1}, s_{j,2}, \ldots, s_{j,end}) \) | a sequence of all station blocks in \( j \) route |
| \( S_{j,j'}^* \) | a sequence of station blocks in \( j \) route without last (two last) elements |
| \( S_{j,j'} \) | a common path of \( j \) and \( j' \), order of blocks such as this of \( j \) |
| \( \rho_j(m), \rho_j(s) \) | subsequent block (station block) in \( j \) route |
| \( \pi_j(m), \pi_j(s) \) | preceding block (station block) in \( j \) route |
| \( l_{\text{out}}(j,s), (l_{\text{n}}(j,s)) \) | time of leaving (entering) the station block \( s \) by the the train \( j \) |
| \( l_{\text{timetable}}(j,s) \) | timetable time of leaving \( s \) by \( j \) |
| \( p_{\text{timetable}}(j, m), p_{\text{min}}(j, m) \) | timetable and minimum time of \( j \) passing \( m \) |
| \( d_{(j,s)} \) | delay of \( j \) leaving \( s \) |
| \( d_{U}(j,s) \) | unavoidable delay of \( j \) leaving \( s \) |
| \( d_{s}(j,s) \) | secondary delay of \( j \) leaving \( s \) |
| \( d_{\text{max}}(j) \) | maximum possible (acceptable) secondary delay for train \( j \) |
| \( \tau_{(1)}(j) \ldots \tau_{(2)}(j) \ldots \) | minimum time for train \( j \) to give way for another train going the same (opposite) direction, |
| \( x_i(\mathbf{x}) \) | binary decision variable (its vector), e.g. \( x_{j,s,d} = x_i \) is 1 if \( j \) leaves \( s \) at delay \( d \) and 0 otherwise, \( i \in \{1, 2, \ldots, n\} \) |
| \( Q \in \mathbb{R}^{n \times n} \) | symmetric QUBO matrix, \( n \) is a number of logical quantum bits. |
| \( f(\mathbf{x}) \) | the objective function, weighted penalty for delays above the unavoidable |

Table 1: Notation summary.
Primary or unavoidable delays (as defined in Section 2.1.1) are denoted by \( d_U(j, s) \). If the already-delayed train enters a railway line \( M \), the initial delay will appear at the first block \( d_U(j, s_{j,1}) \). The unavoidable delay propagates along the line thereby providing a lower bound of delays. Unavoidable delays are required to be non-negative, so we have:

\[
d_U(j, \rho_{j}(s)) = \max\{d_U(j, s) - \alpha(j, s, \rho_{j}(s)), 0\},
\]

(12)

where \( \alpha(j, s, \rho_{j}(s)) \) accounts for the possible time reserve for passing the sequence of blocks, starting from the one directly after \( s \), and ending at the station block \( \rho_{j}(s) \). In the same way, the unavoidable delays are propagated due to the rolling stock circulation minimum times at terminals. Importantly, all unavoidable delays can be computed prior to the optimization.

The secondary delay \( d_S(j, s) \) is denoted as:

\[
d_S(j, s) = d(j, s) - d_U(j, s).
\]

(13)

We assume that the secondary delays have upper bounds \( d_{\text{max}}(j) \). These bounds are the parameters of the model. Their values can either be determined manually (maximum acceptable secondary delays of the given trains), or by using some fast heuristics such as the First Come First Served (FCFS) or First Leave First Served (FLFS) approach (which we discuss later). Setting them too low, however, we will obtain an unfeasible model.

As the upper bound discussed above and the lower bound (the unavoidable delay) are already known:

\[
d_U(j, s) \leq d(j, s) \leq d_U(j, s) + d_{\text{max}}(j)
\]

we can use the (integer) values of the delays as decision variables. The bounds ensure that these variables take their values from a finite range. We shall call this description ‘delay representation’ in what follows; it will be very convenient from the QUBO modeling point of view.

A perfect conflict resolution has no secondary delays. A detailed discussion about the developed objective function is presented in Section 4.1.

### 3.3 Dispatching conditions

Let us have the given train \( j \) and its path \( M_j \) which consists of both station and line blocks. To simplify we assume that the leaving time of the given block equals the entering time on the subsequent block:

\[
t_{\text{out}}(j, m) = t_{\text{in}}(j, \rho_{j}(m)).
\]

(15)

For each train \( j \in J \), each \( m \in M_j \) is assigned two kinds of passing times: a nominal (timetable) \( p_{\text{timetable}}(j, m) \), and a minimum \( p_{\min}(j, m) \). Note that the latter can be smaller due to some reserve, or equal to \( p_{\text{timetable}}(j, m) \).

We address common dispatching conditions, including: the minimum passing time condition, the single block occupation condition, the deadlock condition, the rolling stock circulation condition at the terminal, and the capacity condition.

**Condition 3.1. The minimum passing time condition.** The leaving time from the block section cannot be lower than the sum of the entering time and the minimum passing time.

\[
t_{\text{out}}(j, m) \geq t_{\text{in}}(j, m) + \min(j, m).
\]

(16)

For subsequent station blocks \( s = m_{j,k} \) and \( \rho_{j}(s) = m_{j,1} \):

\[
t_{\text{out}}(j, \rho_{j}(s)) \geq t_{\text{out}}(j, s) + \sum_{i=k+1}^{l} \min(j, m_{j,i}) = t_{\text{out}}(j, s) + \sum_{i=k+1}^{l} p_{\text{timetable}}(j, m_{i}) - \alpha(j, s, \rho_{j}(s))
\]

(17)

where \( \alpha(j, s, \rho_{j}(s)) \) is the time reserve mentioned before. In the delay representation, this condition reads as simple as:

\[
d(j, \rho_{j}(s)) \geq d(j, s) - \alpha(j, s, \rho_{j}(s)).
\]

(18)

(Compare this with (12) where we have an equal sign for the lower limit.)

**Condition 3.2. The single block occupation condition.** Let \( j \) and \( j' \) be two trains heading towards the same direction and sharing their routes between the station \( s \) and the subsequent station \( \rho_{j}(s) \). If the train \( j \) leaves the station block \( s \) at time \( t_{\text{out}}(j, s) \), the subsequent \( t_{\text{out}}(j', s) \geq t_{\text{out}}(j, s) \) train \( j' \) can leave this block at a time for which the following equation is fulfilled:

\[
t_{\text{out}}(j', s) \geq t_{\text{out}}(j, s) + \tau_{11}(j, s, \rho_{j}(s)),
\]

(19)

where \( \tau_{11}(j, s, \rho_{j}(s)) \) is the time required for the train \( j \) to release a way for the train \( j' \) on the route between station block \( s \) and the subsequent station block \( \rho_{j}(s) \). In the delay representation we have:

\[
d(j', s) + t_{\text{timetable}}(j', s) \geq d(j, s) + t_{\text{timetable}}(j, s) + \tau_{11}(j, s, \rho_{j}(s))
\]

(20)
or
\[ d(j', s) \geq d(j, s) + t_{\text{on}}(j', s) - t_{\text{out}}(j, s) + \tau_{11}(j, s, \rho_j(s)). \] (21)

Hence, taking \( \Delta(j, s, j', s) = t_{\text{out}}(j, s) - t_{\text{out}}(j', s) \), we get:
\[ d(j', s) \geq d(j, s) + \Delta(j, s, j', s) + \tau_{11}(j, s, \rho_j(s)). \] (22)

As mentioned before, the condition in (22) needs to be tested for \( t_{\text{out}}(j', s) \geq t_{\text{out}}(j, s) \), i.e. \( d(j', s) \geq d(j, s) + \Delta(j, s, j', s) \), otherwise trains are to be investigated in the reversed order.

The actual form of \( \tau_{11}(j, s, \rho_j(s)) \) depends on the dispatching details of the particular problem. We assume all the time reserves to be realized on stations. In consequence, \( \tau_{11}(j, s, \rho_j(s)) \) is delay independent, what makes the problem tractable.

**Condition 3.3. The deadlock condition.** Assume that two trains \( j \) and \( j' \) are heading towards the opposite directions on a route determined by subsequent station blocks \( s \) and \( \rho_j(s) \) in the path of the train \( j \). In the path of \( j' \) these are reversed, thus \( j \) goes \( s \to \rho_j(s) \), while \( j' \) goes \( \rho_j(s) \to s \). Assume for now that the train \( j \) is to enter the common block section before \( j' \). (The condition has to be checked also in the reverse order.) Let \( \tau_{12}(j, s, \rho_j(s)) \) be the time required for the train \( j \) to get from station block \( s \) to \( \rho_j(s) \). Given this, the deadlock condition states as follows:
\[ t_{\text{out}}(j', \rho_j(s)) \geq t_{\text{in}}(j, \rho_j(s)), \] (23)

that is,
\[ t_{\text{out}}(j', \rho_j(s)) \geq t_{\text{out}}(j, s) + \tau_{12}(j, s, \rho_j(s)). \] (24)

In the delay representation:
\[ d(j', \rho_j(s)) + t_{\text{on}}(j', \rho_j(s)) \geq d(j, s) + t_{\text{on}}(j, s) + \tau_{12}(j, s, \rho_j(s)). \] (25)

and
\[ d(j', \rho_j(s)) \geq d(j, s) + t_{\text{on}}(j', \rho_j(s)) - t_{\text{on}}(j, s) + \tau_{12}(j, s, \rho_j(s)). \] (26)

Hence, taking \( \Delta(j, s, j', \rho_j(s)) = t_{\text{on}}(j', \rho_j(s)) - t_{\text{on}}(j, s) + \tau_{12}(j, s, \rho_j(s)) \), we get:
\[ d(j', \rho_j(s)) \geq d(j, s) + \Delta(j, s, j', \rho_j(s)) + \tau_{12}(j, s, \rho_j(s)). \] (27)

Again, condition (27) needs to be tested for \( t_{\text{out}}(j', \rho_j(s)) \geq t_{\text{out}}(j, s) \), otherwise trains are to be investigated in the reversed order.

Further, analogically to Condition 3.2, the form of \( \tau_{12}(j, s, \rho_j(s)) \) depends on the dispatching details, resulting from the formulation of the problem. Again, assuming having all time reserves at stations, \( \tau_{12}(j, s, \rho_j(s)) \) is delay-independent, what makes the problem is tractable.

As mentioned before, the particular form of \( \tau \)'s are problem dependent. Let us propose the simple approach that was demonstrated to be suitable for aforementioned dispatching situations. Suppose that train \( j \) departs from the station \( s \) to the subsequent station \( \rho_j(s) \), passing the blocks \( m_k, m_{k+1}, \ldots, m_{i-1}, m_i \), where \( s = m_k \) and \( m_l = \rho_j(s) \). The subsequent train proceeding in the same direction is allowed to leave at least at:
\[ \tau_{11}(j, s) = \max_{t \in \{k+1, \ldots, i-1\}} \left( t_{\text{in}}(j, m_{i+1}) - t_{\text{on}}(j, m_i) \right). \] (28)

The subsequent train proceeding in the opposite direction is allowed to leave at least after:
\[ \tau_{12}(j, s) = \sum_{t \in \{k+1, \ldots, i-1\}} \left( t_{\text{in}}(j, m_{i+1}) - t_{\text{on}}(j, m_i) \right) = t_{\text{out}}(j, \rho_j(s)) - t_{\text{out}}(j, s). \] (29)

Referring to the minimum and maximum delay conditions – see (14) – there are such pairs of trains for which either the Condition 3.2, or the Condition 3.3, is always fulfilled. This observation simplifies our QUBO representation of the problem.

**Condition 3.4. Rolling stock circulation condition at the terminal.** If the train \( j \) being operated by a given vehicle (more broadly: the rolling stock) terminates at a station, where \( j' \) starts its course (after turnover) – that is \( s_{j, \text{end}} = s_{j', 1} \), the following condition arises:
\[ t_{\text{out}}(j', s_{j', 1}) > t_{\text{in}}(j, s_{j, \text{end}}) + \Delta(j, j'), \] (30)

where \( \Delta(j, j') \) is the minimum turnover time. In the delay representation we have:
\[ d(j', 1) + t_{\text{on}}(j', 1) > d(j, s_{j, \text{end}-1}) + t_{\text{on}}(j, s_{j, \text{end}-1}) + \tau_{12}(j, s_{j, \text{end}-1}) + \Delta(j, j'). \] (31)

Hence, taking \( R(j, j') = t_{\text{on}}(j', 1) - t_{\text{on}}(j, s_{j, \text{end}-1}) - \tau_{12}(j, s_{j, \text{end}-1}) - \Delta(j, j') \), we get:
\[ d(j', 1) > d(j, s_{j, \text{end}-1}) - R(j, j'). \] (32)
Remark 4.1. Observe that conditions 3.2 - 3.3 (the single block occupation condition and the deadlock condition) refer to the subsequent stations in \( j \)'th train path – \( s \) and \( \rho_j(s) \). (Recall that \( \rho_j(s_{\text{out}}) \) does not exist in our model.) Time of entering of \( \rho_j(s) \) is computed from \( x_{s,j,d} \) and \( \tau_1(j,s,\rho_j(s)) \), but does not refer to \( x_{s,j,d} \). Hence we do not need to investigate the leaving time from the last block of the train’s path. We assume that the arrival time at this block can be computed from the leaving time from the penultimate block and the passing time. (Of course our goal is to reduce the number of QUBO variables in the analysis.) Here, delays at the end of the route are investigated on leaving the penultimate station on analyzed route.

Let \( S_{j,j'} \) be the sequence of blocks being a common route of trains \( j \) and \( j' \). If both these trains are traveling in the same direction, the order of blocks in \( S_{j,j'} \) is straightforward. Alternatively, we need to consider the block sequence of train \( j \) as the reversed sequence of blocks of train \( j' \). Following this, we introduce \( S_{j,j'} = S_{j,j'} \setminus \{s_{y,\text{out}}\} \).
for the conditions 3.2 and 3.3. The condition 3.2 states that two trains traveling in the same direction are not allowed to appear at the same block section. In particular, from (22) it results that:

$$\forall_{(j, j') \in \mathcal{J}^n(\mathcal{T})} \forall_{s \in \mathcal{S}_{j, j'}} \sum_{d \in A_{j, s}} \left( \sum_{d' \in B(d) \cap A_{j', s}} x_{j, s, d} x_{j, s', d'} \right) = 0,$$

(38)

where $B(d) = \{d + \Delta(j, s, j', s), d + \Delta(j, s, j', s) + 1, \ldots, d + \Delta(j, s, j', s) + \tau_1(j, s, \rho_j(s)) - 1\}$ is a set of delays that violates condition 3.2.

Assume now that two trains $j$ and $j'$ are heading in opposite directions. From (27) it results that:

$$\forall_{j \in \mathcal{J}^n(\mathcal{T})} \forall_{j' \in \mathcal{J}^n(\mathcal{T})} \forall_{s \in \mathcal{S}_{j, j'}} \sum_{d \in A_{j, s}} \left( \sum_{d' \in C(d) \cap A_{j', s}} x_{j, s, d} x_{j, s', d'} \right) = 0$$

(39)

where $C(d) = \{d(j, s) + \Delta(j, s, j', \rho_j(s)), d(j, s) + \Delta(j, s, j', \rho_j(s)) + 1, \ldots, d(j, s) + \Delta(j, s, j', \rho_j(s)) + \tau_2(j', \rho_j(s)) - 1\}$.

Referring to Remark 4.1 we do not need to examine delays when leaving the ending station of the train path. For the minimum passing time – condition 3.1 – we introduce $S_{j, s}^{**} = S_j \setminus \{s_{j, end, s_{j, end-1}}\}$. From (18) we have:

$$\forall_{j} \forall_{s \in S_{j, s}^{**}} \sum_{d \in A_{j, s}} \left( \sum_{d' \in D(d) \cap A_{j, \rho_j(s)}} x_{j, s, d} x_{j, \rho_j(s), d'} \right) = 0,$$

(40)

where $D(d) = \{0, 1, \ldots, d - \alpha(j, s, \rho_j(s)) - 1\}$.

Following the the rolling stock circulation condition 3.4, we have from (32):

$$\forall_{j, j' \in \text{terminal pairs}} \sum_{d \in A_{j, \rho_{j, end-1}}} \sum_{d' \in E(d) \cap A_{j', 1}} x_{j, s, (j, s, \rho_{j, end-1})} d \cdot x_{j', s, (j', 1)} d' = 0,$$

(41)

where $E(d) = \{0, 1, \ldots, d - R(j', j)\}$.

The objective of the algorithm is to schedule trains in a way that secondary delays are minimized. The general penalty function for delays can be written in the following way:

$$f(d, j, s) = \hat{f}(\hat{d} j, s),$$

(42)

where $\hat{d} = \frac{d(j, s) - d_l(j, s)}{d_{\text{max}}(j)}$.

As it is discussed in Section 2.1.1, the primary delay ($d_T$) is unavoidable so no penalty is applied for such a delay. The maximum delay above the unavoidable one, is one of the model parameters denoted as $d_{\text{max}}(j)$ – see Eq (14). From this we require $\hat{f}(\hat{d} j, s)$ to fulfill the following:

$$\hat{f}(\hat{d} j, s) = \begin{cases} 0 & \text{if } \hat{d} = 0, \\ \max_{\hat{d} \in [0,1]} \hat{f}(\hat{d} j, s) & \text{if } \hat{d} = 1, \\ \text{is non-decreasing in } \hat{d} & \text{if } \hat{d} \in (0,1). \end{cases}$$

(43)

The bottom case has its roots in the fact, that for higher delays there cannot be lower penalty values. We introduce the linear objective function for penalties calculation for the secondary delays (to be shortened in what follows to just penalty function):

$$f(x) = \sum_{j \in J} \sum_{s \in S_j} \sum_{d \in A_{j, s}} f(d, j, s) \cdot x_{j, s, d},$$

(44)

where $f(d, j, s)$ as weights.

Apart from the constrains discussed above the penalty function can be chosen deliberately, which is a great advantage of the model. By selecting $\hat{f}(\hat{d} j, s)$, one may take advantage of various dispatching policies. This ensures a freedom of choice in striving for the best-suited dispatching solution. Let us mention just a few of them:

1. For a quasi minimization of maximum secondary delays, one may opt for a strongly increasing convex function in $\hat{d}$, such as exponential, geometrical etc.
2. To minimize a number of delayed trains, one may opt for the step function $\hat{d}$.
3. To minimize the sum of delays one may opt for a linear function in $\hat{d}$.
4. Subsequent trains can be assigned various weights for delays on which their priorities depend.
5. A subset of stations can be selected as the only relevant from the point of view of delays. For practical reasons we analyze delays on penultimate stations – see Remark 4.1.

For our particular dispatching problems to be solved, we select policies set out in points 3 – 5.
4.2 QUBO representation: penalty methods

Having our problem formulated as a constrained 0-1 program, we need to make it unconstrained to achieve a QUBO form — see (4). This is usually done with penalty methods [54]. It has been shown in [55] that all binary linear and quadratic programs translate to QUBO along some simple rules.

The problems one faces with a quadratic 0-1 program requires some specific considerations when adopting the penalty method. Let us outline this approach with a focus on our problem. As we have a linear objective function — see (44), it can be written as quadratic, because the decision variables are binary:

$$\min_{x} f(x) = \min_{x} c^T x = \min_{x} x^T \text{diag}(c)x. \quad (45)$$

We need to meet the constraints set out in (38) – (41) to make the solution feasible. These constraints are considered as hard constraints. To achieve an unconstrained problem, we define a penalty function in the following way. We add the magnitude of their violation, multiplied by some well-chosen coefficient, to the objective function.

In particular, we shall have quadratic constraints in the form of:

$$\sum_{(i,j) \in V_p} x_i x_j = 0, \quad (46)$$

excluding pairs of variables which simultaneously are valued 1. We can deal with such a constraint by adding to our objective the following terms:

$$P_{\text{pair}}(x) = p_{\text{pair}} \sum_{(i,j) \in V_p} (x_i x_j + x_j x_i), \quad (47)$$

where $p_{\text{pair}}$ is a positive constant. Additionally, from (37), we have additional hard constraints in the form of:

$$\sum_{i \in V_s} x_i = 1. \quad (48)$$

Out of $x_i$ where $i \in V_s$, one and only one variable $x_i$ is 1. The above-mentioned constrains yield a linear objective function. It may be transformed to the following penalty function:

$$P'_{\text{sum}}(x) = p_{\text{sum}} \left( \sum_{i \in V_s} x_i - 1 \right)^2 \quad (49)$$

that is quadratic. Next we replace the $x_i$-s with $x_i^2$-s in the linear terms, and omit the constant terms as they provide just an offset to the solution. In result, we obtain a quadratic penalty in the form we desire:

$$P_{\text{sum}}(x) = p_{\text{sum}} \left( \sum_{i,j \in V^2, i \neq j} x_i x_j - \sum_{i \in V_s} x_i^2 \right), \quad (50)$$

So our effective QUBO representation is:

$$\min_{x} f'(x) = \min_{x} (f(x) + P_{\text{pair}}(x) + P_{\text{sum}}(x)), \quad (51)$$

which can be written in the form analogical to (4). We shall have many constraints of similar types of Eqs. (46) and (48), thus we have one summand for each in the objective. (It would also be possible to assign a separate coefficient to each of the constraints, though.) Recall that in the theory of penalty methods [54] for continuous optimization it is known that the solution of the unconstrained objective will tend to a feasible optimal solution of the original problem as the multipliers of the penalties ($p_{\text{sum}}$ and $p_{\text{pair}}$ in our case) tend to infinity, provided that the objective function and the penalties obey certain continuity conditions. As in our case both the objective and the penalties are quadratic, this convergence would be warranted for the continuous relaxation of the problem. And even though we have a 0-1 problem, if we had an infinitely precise solution of the QUBO at hand, increasing the parameters would result in a convergence to a feasible optimum.

However, somewhat analogously to the continuous case (where the Hessian of the unconstrained problem diverges as the parameters grow, making the unconstrained problem numerically ill-conditioned), the properties of the actual computing approach or devices makes it more cumbersome to make a good choice of multipliers.

In particular, recall that our actual solution for the unconstrained problem will be the lowest eigenvalue and a corresponding eigenvector of a symmetric matrix, and the eigenvector itself is actually the energy of a (real or model) physical system, which is in a finite range. The parameters $p_{\text{sum}}$ and $p_{\text{pair}}$ have to be chosen so that the terms representing the constraints in this energy do not dominate the original objective function. If the penalties are too high, the objective is just too small of a perturbation on it, which will be lost in the noise of the physical quantum computer or the numerical errors of an algorithm modeling it. If, however, the penalty coefficients are too low, we get infeasible solutions.

13
The multipliers can be assigned in an ad hoc manner by experimenting with the solution, however, a systematic, possibly problem-dependent approach for their appropriate assignment, (like in case of classical penalty methods, c.f. [54]) would be very desirable in order to make the QUBO optimization more reliable and prevalent.

To get some hint to find the way of how to determine the coefficients of the summands that warrant feasibility, let us first consider a direct search solution of a QUBO of the form in (4). This amounts to evaluating the objective function with all possible values of the decision variables. In our actual effective QUBO in (51), the total matrix $Q$ is a sum of the terms in Eqs. (47) and (50), and the original objective function of (45). So we are facing a sum of three QUBOs, and the objective function value is actually linear in the matrix of QUBOs. Hence, the objective value will be the sum of the original objective function value, and the actual values of the summands representing the constraints.

The feasibility terms have a negative minimum $-L$ because of the omitted 0th order terms when using (50) (instead of (49) achieved if the solution is feasible). For each such condition the value $p_{\text{sum}}$ is contributed to $L$, hence $L = p_{\text{sum}} \cdot \text{(number of linear constrains)}$. The value:

$$f''(x) = P_{\text{pair}} + P_{\text{sum}} - L$$

will be zero if the solution is feasible, and non-zero otherwise. We will call it the hard constraints’ penalty. Recall that we also defined the terms of “soft penalty” not in the auxiliary sense of the penalty methods, but as a penalty value for secondary delays in the railway problem. Hence our objective function can be interpreted as a penalty for violating the “soft constraints”. These are violated if the secondary delays exist. We expect the objective – a kind of “soft constraint penalty” – to be non-zero and to be minimized during the optimization procedure.

If there is an unfeasible minimum where the “cost” of violating some hard constraints is lower than the particular objective function (i.e. the “cost” of violating soft constrains), the effective QUBO may yield an unfeasible minimum. A way to avoid this is to ensure that the lowest violation of any hard constraint have a larger contribution to the objective function (i.e. the “cost” of violating soft constrains), the effective QUBO may yield an unfeasible minimum.

A way to avoid this is to ensure that the lowest violation of any hard constraint have a larger contribution to the objective function, than a violation of all soft constrains of a given feasible (not necessarily optimal) solution. Such a solution can be obtained by some fast heuristics. By having achieved that whenever a constraint is violated, there is a feasible optimum which has a lower effective objective value.

This suggests that one should assign high coefficients to the hard constraints. If one employs a direct search algorithm calculating the values of the objective very accurately, this approach can work out easily. However, the numerical accuracy is always limited, and other inaccuracies of the minimum search can also appear. In case of a physical quantum computer, this is due to the noise of the system. What we get in reality is not the warranted absolute minimum, but a set of samples: vectors for which the effective objective function is close to the minimum. If the coefficients are too high, the original objective function is just a small perturbation over the feasibility violations. Hence, while obtaining strictly feasible solutions, the actual minimum can be lost in the noise.

In a physical system, there should be an upper bound of the effective objective function values that can be obtained. Let us call this bound $R$, thus if the original objective function is positive semidefinite, we shall obtain effective objective function values in $[-L, R]$. We can assume that the feasible optimum of the effective objective function is in this range, as the QUBO will be mapped into this range anyway. The coefficients should be chosen so that the sum of the feasible optimum (which we do not know, so it should be estimated) and the “price” of a few hard constraint violations are still below $R$. This in fact contradicts with the previous requirement of having huge penalties for them. So finding the appropriate values of $p_{\text{sum}}$ and $P_{\text{pair}}$ amounts to the values which address both of the criteria to a suitable extent.

4.3 A simple example

Before moving to numerical studies, we demonstrate our approach on a simple example to illustrate the matter. Consider two trains $j \in \{1, 2\}$, two stations $s \in \{1, 2\}$ and a single track between them. The passing time value (scheduled and minimum) between the stations is 1 (minute) for both trains. A train $j = 1$ is ready to depart from station $s = 1$ (heading to $s = 2$) at the same time as a train $j = 2$ is ready to depart from station $s = 2$ (heading to $s = 1$). If so, a conflict appears on a single track between the stations.

Let the initial delay of both trains be $d = d_U = 1$. As one of the trains needs to wait a minute to meet and pass another one, the maximum acceptable secondary delay is set as $d_{\text{max}} = 1$, see (35). Taking the QUBO representation as in (36) (that is $x_{s,j,d}$) we have the following 4 quantum bits: $x_{1,1,1}, x_{1,1,2}$ (train 1 can leave station 1 at delay 1 or 2), $x_{2,2,1}$ and $x_{2,2,2}$ (train 2 can leave station 2 at delay 1 or 2). The linear constraints say that each train departs from each station once and only once, so (37) takes the form:

$$x_{1,1,1} + x_{1,1,2} = 1 \text{ and } x_{2,2,1} + x_{2,2,2} = 1.$$ (53)

Referring to (50), the optimization sub-problem is as follows:

$$P_{\text{sum}} = -p_{\text{sum}} \left( x_{1,1,1}^2 + x_{1,1,2}^2 - x_{1,1,1}x_{1,1,2} - x_{1,1,2}x_{1,1,1} + x_{2,2,1}^2 + x_{2,2,2}^2 - x_{2,2,1}x_{2,2,2} - x_{2,2,2}x_{2,2,1} \right),$$ (54)

with the optimal value equals to $-L = -2p_{\text{sum}}$. 

14
The quadratic constraint is that two trains are not allowed to depart from the stations at the same time, that is: \(x_{1,1,1} + x_{2,2,1} = 0\) and \(x_{1,1,2} + x_{2,2,2} = 0\). Using (47) the optimization subproblem takes the form:

\[
P_{\text{pair}} = p_{\text{pair}} (x_{1,1,1}x_{2,2,1} + x_{2,2,1}x_{1,1,1} + x_{1,1,2}x_{2,2,2} + x_{2,2,2}x_{1,1,2}),
\]

(55)

with the optimal value equals to 0. Note, that since we have only two stations in this simple example, the minimum passing time condition does not appear \((S^* = 0)\).

Finally, a possible objective function is:

\[
f(x) = x_{1,1,2}w_1 + x_{2,2,2}w_2 = x_{1,1,2}^2w_1 + x_{2,2,2}^2w_2.
\]

(56)

where the secondary delay of train 1 is penalized by \(w_1\) and the secondary delay of train 2 is penalized by \(w_2\).

Let us denote the vector of decision variables by \(x = [x_{1,1,1}, x_{1,1,2}, x_{2,2,1}, x_{2,2,2}]^T\). The QUBO problem can thus be written in the form of (4), here:

\[
Q = \begin{bmatrix}
-p_{\text{sum}} & p_{\text{sum}} & p_{\text{pair}} & 0 \\
p_{\text{sum}} & -p_{\text{sum}} + w_1 & 0 & p_{\text{pair}} \\
p_{\text{pair}} & 0 & -p_{\text{sum}} & p_{\text{sum}} \\
0 & p_{\text{pair}} & p_{\text{sum}} & -p_{\text{sum}} + w_2
\end{bmatrix}.
\]

(57)

As the solution is parameter-dependent, we can use various trains prioritization policy. For the sake of demonstration, assume that the train \(j = 2\) is assigned higher priority than the train \(j = 1\). This implies the assignment of different weights for penalties. We set \(w_1 = 0.5\) and \(w_2 = 1\).

As discussed in Section 4.2, to ensure the calculated solution is feasible, we require to meet the following conditions: \(p_{\text{sum}} > \max\{w_1, w_2\}\) and \(p_{\text{pair}} > \max\{w_1, w_2\}\). We propose \(p_{\text{pair}} = p_{\text{sum}} = 1.75\), making the matrix \(Q\) to take the following form:

\[
Q = \begin{bmatrix}
-1.75 & 1.75 & 1.75 & 0 \\
1.75 & -1.25 & 0 & 1.75 \\
1.75 & 0 & -1.75 & 1.75 \\
0 & 1.75 & 1.75 & -0.75
\end{bmatrix}.
\]

(58)

The optimal solution is \(x = [0, 1, 1, 0]^T\) (train 2 goes first) with \(f'(x) = -3\). Another feasible solution (not optimal, though) is \(x = [1, 0, 0, 1]^T\) (train 1 go first) with \(f'(x) = -2.5\). Other solutions are not feasible: for example \(x = [1, 0, 1, 0]^T\) is not feasible as these two trains are expected to depart from the stations at the same time, here \(f'(x) = 0\). Observe that the classical heuristics (such as FCFS, FLFS) do not make a difference between the two feasible solutions, as both trains enter the conflict segment in the same time and need the same time to pass it. Also both the solutions involve the same value of the secondary delay.

Having formulated our model as a QUOBO problem, it is ready to be solved on a physical or simulated quantum annealer.

### 5 Numerical studies

In this section, we discuss some possible situations of train dispatching on the railway lines managed by the Polish state-owned infrastructure manager PKP Polskie Linie Kolejowe S.A. (PKP PLK in what follows). In particular, we consider two single-track railway lines:

- Railway line No. 216 (Nidzica – Olsztyn section)
- Railway line No. 191 (Goleszów – Wisła Uzdrowisko section).

Railway line No. 216 is of a national importance. It is a single-track section of a passenger corridor Warsaw - Olsztyn, which has been modernized recently. There are both Inter-City (IC) and regional trains operating on the Nidzica – Olsztyn section of line No. 216. In this paper we consider an official train schedule (as for April, 2020). The purpose of the analysis of this section is to demonstrate the simple application of our methodology on a real-life railway section.

Railway line No. 191 is of local importance. The main train service on 191 railway line is Katowice – Wisła Głębce, operated by a local government-owned company ‘Koleje Śląskie’ (eng. Silesian Railways; abbr KS). There are a few Inter-City trains of higher priority there as well. Since 2020 the traffic at this section is suspended due to comprehensive renovation works (there is a temporary rail replacement bus service instead). In our paper, we refer to the parameters of the line after its commissioning, based on public procurement documents [56]. On the basis of these parameters, a cyclic timetable was created. The aim of analyzing this case is to show broader application possibilities of the methodology.

Below we discuss some schemes and train timetables of the above-mentioned railway lines.
5.1 Description of railway lines

In Figure 2(a) we present a segment of the railway line No. 216 (Nidzica – Olsztynek section) and in Figure 2(b) we present the actual timetable in the form of a train diagram is depicted.

In Figure 2(a), 3 stations are presented. Block 1 represents Nidzica station which has two platform edges numbered according to the rules of PKP PLK. Block 3 represents Waplewo station with another two platform edges. Olsztynek station with three platform edges is represented by block 5. The model involves two line blocks with the labels 2 and 4. It is assumed that it takes the same amount of time to get through a given station block and that track the train uses. To leave the station it is required to have the subsequent block free.

As of the trains, Figure 2(b) represents their planned paths. There are 3 trains modeled: two Inter-City ones in red, and a regional train in black. The scheduled meet and pass situations takes place in Waplewo and Olsztynek (which may be a subject of change in the case of a delay). IC5320 leaves the station block 5 (Olsztynek) at 13:54, has a scheduled stopover at the station block 3 (Waplewo) from 14:02 to 14:10 to meet and pass IC5321, and finally arrives at station block 1 (Nidzica) at 14:25. As of the opposite direction, IC3521 leaves the station block 1 at 13:53, stops at the station block 3 from 14:08 to 14:10 and arrives at station block 5 (Olsztynek) at 14:18. These two trains depart at the same time from station block 3 in opposite directions. The third train considered is R00602. It is scheduled to leave block 5 at 14:20, stops at station block 3 (Waplewo) from 14:29 to 14:30, so it is scheduled to start occupying this track 19 minutes after both ICs left. It is behind the IC5320 during the whole section, and does not meet the IC train at all, thus the original schedule is feasible and conflict-free.

Now let us add a 15 minute delay to the departure time of the IC5320 train from station block 5, and 5 minutes to that of the IC5321 from the station block 1. The passing times were originally scheduled according to the maximum permissible speeds. The minimum waiting times at all the considered times are 1 minute regardless of the train type. This introduces the following situation: the two Inter-City trains and the regional train have a conflict at line block 4. This schedule will be referred to as the “conflicted diagram” – see Fig 3(a). The resolution of this conflict requires taking a decision at station blocks 3 and 5.

Let us now turn our attention to the other example. The line segment is presented in Figure 2(c), in turn, another segment is presented (a part of the railway line No. 191), whereas the considered train paths of the real timetable are shown in Figure 2(d). There are 4 stations and another 3 stops for passengers modeled. Here, block 1 represents Goleśzów station and Ustroń station (which has two platform edges and is represented by block 3). Subsequently, we have three line blocks numbered 4, 5, and 6, with two stops for passengers: Ustroń Zdrój and Ustroń Poniewie (with one platform edge each). Next, we have a station block 7 – Ustroń Polana – which has two platform edges. Between the latter and Wisła Uzdrowisko station (numbered 10, three platform edges), there are two more line blocks (8 and 9) with one stop for passengers (Wisła Jawornik). We also assume that it takes exactly the same time to get through the block, regardless of track used – the only what is required is to have the subsequent block free (as in the previous case).

There are 6 trains, two Inter-City ones in red and 4 regional (KS) in black presented in Figure 2(d). Regional trains serve all the stops and stations, while Inter-City service stops only at stations. We consider Wisła Uzdrowisko (station block 10) to be a terminus for the Inter-City trains (however, it does not apply to the regional trains which go further on). In this situation, there are no meet and pass situations at intermediate stations (Ustroń and Ustroń Polana) in the original timetable. All the Inter-City trains are assumed to have the same rolling stock circulation scheme and the minimum service time of $R(j,j') = 20$ minutes at the terminus for ICs (block 10), c.f. Condition 3.4.

We analyze the following dispatching cases, selected to present the algorithm behavior in various situations:

1. A moderate delay of Inter-City train setting off from the station block 1, see Fig. 11(a);
2. A moderate delay of all trains setting off from station block 1, see Fig. 11(b);
3. A significant delay of some trains setting off from the station block 1, see Fig. 11(c);
4. A serious delay of the Inter-City train setting off from the station block 1, see Fig. 11(d).

The conflicted timetables of cases 1 – 4 are presented in Figure 11.

5.2 Conventional approach: simple heuristics

As simple conventional approaches serving as a comparison, and also as a simple way of obtaining reasonable feasible solutions, we use the standard heuristics: the FCFS (First Come First Served), the FLFS (First Leave First Served) and the AMCC (Avoid Maximum Current $C_{max}$) [19], where $C_{max}$ is consistent with the maximum secondary delay of the partial problem. All these heuristics are used to determine the order of trains when passing the blocks (for the implementation reason the trains are analyzed in pairs). The FCFS and FLFS are quite simple and they are rather common in the real-life train dispatching around the world. In these heuristics the way is given to the train that first arrives – or first leaves – the analyzed block section. In practice, the decisions based on both these heuristics are taken starting from the most urgent conflict. Next, since passing and overtaking is only possible at stations, so-called implied selections [20] are determined. The procedure is repeated as long as all the conflicts are solved.
The AMCC is a more complex approach which objective is to minimize the maximum secondary delay of the trains; this objective will be referred to as the “AMCC objective” in what follows. This is quite an intuitive procedure, yet more sophisticated than FCFS and FLFS. In support of the comparison, stations are assigned an infinite capacity. Of course, solutions requiring a capacity higher than that of the given station have to be rejected.

Referring to the railway line presented in Fig. 2(a): for the conflicted timetable in Fig. 3(a), each heuristics return the same solution presented in Fig. 3(b). When comparing the FCFS with the FLFS heuristics, observe that in the conflicted timetable, 3 trains (IC5320, IC5321, R90602) are scheduled to occupy block 4 simultaneously, which is forbidden.

To avoid the conflict, IC5321 is allowed to enter this block with a 3 minute delay at 14:17 (as soon as IC5320 leaves it), thus leaving the block at 14:25 instead of 14:22, which results in 3 minutes of secondary delay. Consequently, R9062 is allowed to enter the block at 14:25 earliest, an additional 4-minute delay as compared to the conflict timetable. Thus the maximum secondary delay is 4, and the sum of delays at entering last blocks is 7. The maximum secondary delay equals to 4 minutes, it is the lowest possible one, hence the solution is optimal with respect to the AMCC objective.

Another example – presented in Fig. 2(c) – is more complex, yet it is still solvable by the state-of-the-art quantum computing machine. We do not discuss this example in details, but we only refer to the maximum secondary delays as the objective function. This is presented in Tab. 2 for the discussed heuristics. The upper limit used in the quantum computing is set to $d_{max} = 10$. (Observe that most of the secondary delay values are within this limit.) All the train diagrams are presented in the Appendix A, Figs. 12 13 14. The values of the AMCC objective function are presented in Tab. 2; AMCC appears to find the actual optimum in these cases, thus providing a good enough reference for comparison, albeit with an objective function different from that of ours. Our choice of the objective will be more flexible, thus will still leave room for further nontrivial optimization.
5.3 Emulated and quantum QUBO solution

Our approach based on quadratic unconstrained binary optimization (QUBO) concerns the objective function set out in (51). This contains the feasibility conditions (hard constrains) and the objective function \( f(x) \) (soft constrains) of (59). For the feasibility part we need to determine: \( \tau_{(1)}(j,s) \) - the minimum time for the train \( j \) to give way for another train going in the same direction, and \( \tau_{(2)}(j,s) \) - the minimum time for train \( j \) to give way for the other train going in the opposite direction (see condition 3.2 and condition 3.3).

In as opposed to the previously described conventional approaches, the QUBO objective function introduces flexibility in choosing the dispatching policy by setting the values of the penalty weights for the delays of trains. In this way almost any train prioritization is possible. To demonstrate this flexibility we make the penalty values proportional to the secondary delays of the trains which enter the last station block. This is equivalent to the secondary delay at leaving the penultimate station block. Each train is assigned a weight \( w_j \), yielding the form of (44):

\[
f(x) = \sum_{j \in J} \left( \sum_{d \in A_{j,s^*}} w_j \cdot \frac{d(j,s^*) - d_U(j,s^*)}{d_{\max}(j)} \cdot x_{j,s^*,d} \right),
\]

where \( s^* = s_{(j,e_{nd-1})} \).

The following train prioritization is adopted. In the case of railway line No. 216, the Inter-City trains are assumed to have higher value of the delay penalty weight \( w = 1.5 \), while the regional one has is weighted \( w = 1.0 \). We give higher priority value for the Inter-City train, which is quite a reasonable approach, resulting from the train prioritization in Poland (and in many other countries). In the another case (line No. 191), the priorities of trains heading towards block 10 (Wisła Uzdrowisko) are lower, weighted 0.9 for all the trains. However, train priorities for the trains heading in the opposite direction (towards block 1 – Goleśzów – and beyond the analyzed section) have higher values: 1.0 for the regional trains and 1.5 for Inter-Cities. Such a policy is motivated by the unwillingness to propagate the delays across the Polish railway network – that regional trains proceeds towards the main railway junction in the region’s major city (Katowice), and Inter-City train service is scheduled towards the state capital city (Warsaw). Observe that \( w_j \) is the highest possible penalty for delay of train \( j \); see (59). In both these cases the maximum of \( w_j \) is 1.5. Hence, the penalties for non-feasible solution should be higher, for more detailed discussion see Section 4.2. We set \( p_{\text{pair}} = p_{\text{sum}} = 1.75 > 1.5 \).

Referring to (14), we have the maximum secondary delay \( d_{\max} \) parameter (for simplicity we assume that \( d_{\max} \) is the same for all trains and all analyzed station blocks). For sure, it cannot be smaller than the one returned from the AMCC heuristics. However, since the AMCC may not be optimal in terms of our objective function, we need to
leave a margin for some greater values of maximum secondary delay. On the other hand, since the system size grows with the $d_{\text{max}}$, it must be limited enough to make the problem applicable for state-of-the-art quantum devices and classical algorithms motivated by them. In details, since we do not analyze the delays on the last station of the analyzed segment on the line, we require as many as $(\text{number of station blocks} - 1) \cdot (\text{number of trains}) \cdot (d_{\text{max}} + 1)$ qubits.

In case of the railway line No. 216, the $d_{\text{max}} = 7$, which is considerably larger than the AMCC solution. To handle the problem, we require 48 quantum bits. This makes the problem suitable for both the quantum computing at the current state of art, and the GPU-based implementation of the brute-force states (ground and subsequent excited) [53]. These approaches require the limit of 50 quantum bits.

There are much more dispatching solutions in case of the line No. 191, what makes the analysis more interesting from the dispatching point of view. We set $d_{\text{max}} = 10$, for justification see Tab. 2 and observe that $d_{\text{max}}$ is considerably larger than the AMCC output. The $d_{\text{max}} = 10$ yields 198 logical quantum bits, which we were able to embed on present-day D-Wave device DW-2000Q, in most cases. Current quantum annealing devices are imperfect and often output excited states (rather then the ground one). Amongst sources of error, one could list noise and lack of precision of the coefficients of the Ising problem resulting from the Digital to Analog Converter (DAC) quantization.

The clue of our approach comes from the fact, that the excited states still provide the optimal dispatching solutions provided their corresponding energies are relatively small. This is for the reason that what is really to be determined is the order of trains leaving from each station block (i.e. this is the decision to be made). Crucial here is to determine all the “meet and pass”, and the “meet and overtake” situations. Exact time of leaving block sections is determined is the order of trains leaving from each station block (i.e. this is the decision to be made). Crucial here is to determine all the “meet and pass”, and the “meet and overtake” situations. Exact time of leaving block sections is determined.

5.3.1 Exact calculation of the spectrum

To demonstrate the aforementioned idea, we first present the results of brute-force numerical simulations performed on a GPU architecture [53]. With this approach, the low-energy spectrum for smaller instances considered in this work have been calculated exactly, providing some guidance for the understanding of the model behavior and parameter dependence. The method is suitable for small (i.e., up to $N \leq 50$ quantum bits), but otherwise arbitrary systems. To study the impact of the (hard) penalties for non-feasible solution, apart from $p_{\text{pair}} = p_{\text{sum}} = 1.75$ in (51), we use another, higher penalties and not equal one to another, $p_{\text{pair}} = 2.7$, and $p_{\text{sum}} = 2.2$.

Let us assume that the solution in Fig. 3(b) is the optimal one. Here the train IC3521 ($w = 1.5$) waits for 3 minutes at block 3, while the regional one R90602 ($w = 1.0$) waits for 4 minutes at block 5, causing 4 minutes of the secondary delay while leaving block 3. This gives the penalty 1.214. Referring to the feasibility terms in (51), if the solution is feasible $P'_{\text{sum}} = 0$, the linear constrain gives the negative offset to the energy. Referring to (50) as we have 3 trains for which we analyze 2 stations, this negative offset is $P'_{\text{sum}} = -3 \cdot 2 \cdot p_{\text{pair}}$. Based on the feasibility terms set out (51) yields $-10.5$ for $p_{\text{sum}} = 1.75$, and $-13.2$ for $p_{\text{sum}} = 2.2$. This returns the ground state energy of $f'(x) = -9.286$ and $f'(x) = -11.986$ accordingly. Finally, in the ground state solution shown in Fig. 3(b), the IC3521 train can leave the station block 1 with a secondary delay of 0, 1, 2, 3, not affecting any delays of the trains leaving block 3. All of these situations lead to the ground state energy. Hence our approach produces the degeneracy of the ground state equal to 4 in this case.

Spectra of solutions and their degeneracy are presented in Fig. 4(b) and Fig. 4(a). All the solutions equivalent to the ground state from the dispatching point of view, are marked in green. Non-feasible excited states solutions

Figure 4: Spectra of the “low energy” solutions for two penalties strategies from the brute-force (exact) solution. The black line separates the phase where only feasible solutions appear. Observe the mixing phase where both feasible and not feasible solutions occur.
Figure 5: Embedding of a simple, six-qubit problem. Left: graph of the original problem. Right: problem embedded into a unit cell of Chimera. Here, different colors correspond to different logical variables. Apparently the original problem does not map directly into Chimera as it contains cycles of length 3. Therefore, two chains have to be introduced. Couplings corresponding to inner-chain penalties are marked with the same color as the variable they correspond to.

(where some of the feasibility conditions set out in (51) are broken), are marked in red. In this example we do not have feasible solutions which are not optimal, i.e. where the order of trains at a station is different from the one in the ground state solution.

In the case of the line No. 191 a more detailed analysis of the spectrum of solutions was possible due to the generality of the brute-force simulation. The results are presented in Fig. 4. We shall find later on that the actual D-Wave solutions managed to get into the “green” tail of feasible solutions, but high degeneracy of higher energy states may impose some risk of ending up in more frequently appearing excited states c.f. Fig. 6.

5.3.2 Classical heuristic approach

By means of classical simulations we hope to achieve the ground state of the (51) or at least low excited states equivalent to the ground state with respect to the dispatching problem. It is important to mention that hereafter we transform the original QUBO coding onto the Chimera graph coding - see section 2.2.1. This makes the algorithm ready for processing on the real quantum annealer. As for a simple example of the embedding we refer to the problem of size 4 quantum bits that has been discussed in Section 4.3. In that case the mapping was trivial. In case of 6 quantum bits, for instance (by setting $d_{\text{max}} = 2$), we will have additional terms in (54), (55), and (56). Hence, the larger problem cannot be directly mapped on the Chimera graph, so the embedding procedure is required, as illustrated in Fig. 5. This illustrates the basic idea of how the embedding is performed in even larger models.

As of the model parameters, recall that for the particular QUBO we have opted for $p_{\text{pair}} = p_{\text{sum}} = 1.75$ or $p_{\text{pair}} = 2.2$ $p_{\text{sum}} = 2.7$ for the line No. 216 and $p_{\text{pair}} = p_{\text{sum}} = 1.75$ only for the line No. 191. Let us present the solutions resulting from the two state-of-the-art numerical methods, which we shall later compare with the experimental results obtained by running the D-Wave 2000Q quantum annealers. The first solver is developed ‘in-house’ and is based on tensor network techniques [51]. The idea behind this solver is to represent the probability of finding a given configuration by a quantum annealing processor as a PEPS tensor network [51]. This allows, in particular, for an efficient bound and branch strategy to be applied in order to find $M \ll 2^N$ candidates for the low energy states, where $N$ is the number of physical quantum bits on the Chimera graph. In principle, such heuristic should work well for rather simple QUBO problems i.e. such where the $Q$ matrix in (4) has some identical or zero terms, this corresponds to the so called weak entanglement regime. One can show that this is the case in our problem (see also the simple example of the $Q$ matrix in (57)). Furthermore, heuristic parameters such as the Boltzmann temperature ($\beta$) can be provided, allowing to ‘zoom in’ on the low energy spectrum depending on the problem in question. We set $\beta = 4$, which is quite a typical setting as discussed in [51]. Although even better solutions may potentially be achieved by tuning this parameter up, we demonstrated that this default setting is satisfactory form the dispatching point of view. The second classical solver is CPLEX [57] (version 12.9.0.0) which is one of the best commercially available optimization software provided by IBM. In our work we used DOcplex Mathematical Programming package (DOcplex.MP) for Python.

Concerning the results about the another railway line (No. 191) the values of the objective function (59) are tabulated Tab. 3. We also include the values of our objective function for the FLFS, FCFS and the AMCC-optimal solutions. The slight advantage of AMCC and FCFS over the CPLEX or the tensor network in the case 2 is caused by the fact that the (28) used in the QUBO construction is an approximation and in the block to block analysis the regional train Ks47 might have run a bit earlier, following the IC2 train. This, however, does not affect the optimality of the solution form the dispatching point of view.

The CPLEX results in most cases refer to the ground state of the QUBO. We are interested in the results being equivalent with those of CPLEX from the dispatching point of view. These results are marked in blue in Tab. 3. The tensor network approach gives equivalent solutions to those of CPLEX. However, the tensor network
Table 3: The values of the objective function $f(x)$ for the solutions obtained from the classical calculation of the QUBO and those from all the heuristics. The blue color represents equivalence from the dispatching point of view with the the ground state of the QUBO (the same order of trains at each station). For each of the heuristics there is the case in which it does not produce the QUBO-optimal solution.

| Method       | case 1 | case 2 | case 3 | case 4 |
|--------------|--------|--------|--------|--------|
| CPLEX        | 0.54   | 1.40   | 0.73   | 0.20   |
| Tensor Network | 0.54   | 1.40   | 1.65   | 0.29   |
| AMCC         | 0.77   | 1.30   | 0.73   | 0.20   |
| FLFS         | 0.54   | 1.71   | 0.73   | 0.20   |
| FCFS         | 0.77   | 1.30   | 0.95   | 0.20   |

Figure 6: Distribution of energies corresponding to states (solutions), which are sampled by D-Wave 2000Q quantum annealer. In particular, 1000 samples were taken for each annealing time, and the strength of embedding was set to $c_{ss} = 2.0$. This device is still very noisy and prone to errors. Thus, the low-energy spectrum is shifted towards higher energies. Ideally it should be centered around the ground states (global optimum).

5.3.3 Quantum computing on the D-Wave machine

The solver we discuss (D-Wave 2000Q annealer) is of a probabilistic nature. In particular, the output is a sample from the low-energy spectrum. The solution is thus assumed to be the element of this sample with the lowest energy (in practice, these are not from the ground state but one hopes for the low excited state). Hence, the more runs on the D-Wave machine the lower the energy may appear.

As already mentioned, qubits on the D-Wave’s chip are arranged into a Chimera graph topology. Furthermore, some nodes and edges may be missing on the physical device, making the topology different even from an ideal Chimera graph. This leads to the need of minor embedding of the problem, mapping logical qubits to physical
ones. To this end, multiple physical qubits are chained together to represent a single logical variable, which increases their connectivity at the cost of number of available qubits. Such embedding is performed by introducing a penalty term that favors states in which quantum bits in each chain are aligned in the same direction. (Note that we encounter yet another penalty at this point.) The multiplicative factor governing this process is called chain strength, and it should dominate all coefficients present in the original problem. In this work, we set this factor to a maximum absolute value of the coefficients of the original problem multiples by a parameter which we call chain strength scale (css). In our experiment, css is ranged from 2.0 to 9.0. Another parameter is the annealing time (ranging form 5µs to 2000µs). This is the actual duration of the physical annealing process.

In Figs. 7(c) 7(d) we present energies of the best outcomes of D-Wave machine for line No. 216 and various annealing times. The green dots represent the feasible solutions (and equivalent with the optimal one), while the red dots represent solutions that are not feasible. In general the quality of solution faintly rises with the annealing time, however it will be seen on large examples that the best results are for time somewhere between 1000 and 2000. What is rather unusual observation is that more feasible solutions were find in the lower hard penalties constraint case \( p_{\text{sum}} = p_{\text{pair}} = 1.75 \). Such penalty setting will be the reference for the large examples investigations. Here the embedding strength css = 2 was selected, i.e. the lowest possible. This is a good choice as demonstrated later in Fig. 8. Best solutions D-Wave are presented in the form of train diagrams in Figs. 7(a) 7(b).

The quality of solutions in relation to the css strength for various parameter settings is presented in Fig. 8. We have observed that in our cases the quality of the solution degraded with increase of css. This is unusual, as increasing css typically yields more solutions without broken chains that do not need to be post-processed to obtain a feasible solution of the original problem. This may be caused by the fact, that the large coupling of the embedding, may cause the problem constrains to be just a little perturbation in the physical QUBO. These, as discussed earlier may be hidden in the noise of the D-Wave 2000Q annealer.

Hence we select css = 2.0 (the lowest possible value) for further investigation. Some examples of the hard and soft constraints values are presented in Tab. 4. Again, it appears that the higher the values of \( p_{\text{sum}} \) and \( p_{\text{pair}} \), the higher values of \( f(x) \) are observed. This may be caused by soft constraints lost in the noise of the D-Wave 2000Q annealer.

For the railway line No. 191 finding the feasible solution is more difficult. Hence we took advantage of the maximal number of runs on the D-Wave machine, equals to 250 000 runs. The results of the lowest energies and penalties are presented in Fig. 9. We had to skip case 3 for the higher number of feasibility constrains leading to the failure to find any embedding on a real Chimera. Interestingly though, recall that we had found the embedding for the ideal Chimera while simulating the solution, c.f. Section 5.3.2. Hence, the failure in case of the real graph

Figure 7: Train diagrams of the best D-Wave solutions, the energies of D-Wave solutions (green: feasible, red: not feasible) and the optimal tensor network solution.
Figure 8: Line No. 216, the minimal energy from the D-Wave, 1000 runs. Green dots indicates the feasible solutions, while red the unfeasible ones. In general the energy rises as the css rises. We do not see evidence that the different setting of $p_{\text{pair}}$ and $p_{\text{sum}}$ improve the feasibility, see Fig. 8(b).

Figure 9: Line No. 191, the minimal energy from the D-Wave at 250k runs, css = 2.0 was selected and $p_{\text{pair}} = 1.75, p_{\text{sum}} = 1.75$. The output is rather weakly dependent on the annealing time and still far from the ground state.

is possibly caused by the fact that some of the required connections or nodes are missing from the real Chimera. Finding the feasible solution in such a case (while having non-zero hard constrains penalties) is a problem for further research. One would expect that increasing $p_{\text{pair}}$ and $p_{\text{sum}}$ parameters can be helpful. However, it may aggravate the soft constraints’ penalties to an ever greater degree. From Fig. 9(b), the penalties of soft constraints – the values of the objective function $f(x)$ – are much higher than the optimal one presented in Tab. 3. Although the solutions are not feasible, we select the two where only one hard constrain is violated ($f'(x) = 1.75$), these are: case 1 and 1400µs of the annealing time, and case 2 and 1200µs of the annealing time. Train diagrams of these solutions are presented in Fig. 10. Note that both these diagrams can easily be modified by the dispatcher to obtain a feasible solution. The case in Fig. 10(a) can be amended by introducing the lacking 1 minute of stay for Ks3 in station 7. Here the solution would not be optimal, and would be different from the optimal one got from CPLEX, Tensor Network and FLFS, and also from other non optimal, yet feasible, returned by FCFS and AMCC. The case in Fig. 10(b) can be upgraded by shortening the stays of Ks3 and IC2 and letting them meet and pass at station 10. Here the solution would be optimal and equivalent to this of the tensor network and CPLEX.

| css | $p_{\text{sum}},p_{\text{pair}}$ | hard constrains penalty $f'(x)$ | $f(x)$ |
|-----|------------------|------------------|-------|
| 2   | 1.75, 1.75       | 0.0              | 1.36  |
| 2   | 2.2, 2.7         | 0.0              | 1.57  |
| 4   | 1.75, 1.75       | 0.0              | 1.93  |
| 4   | 2.2, 2.7         | 2.2              | 2.07  |
| 6   | 1.75, 1.75       | 5.25             | 0.43  |
| 6   | 2.2, 2.7         | 6.6              | 0.86  |

Table 4: Line No. 216, objective functions and penalties for violating hard constrains, c.f. (52). Output from D-Wave, at 2000µs of the annealing time. If $f'(x) > 0$, the solution is not feasible. The $p_{\text{sum}} = p_{\text{pair}} = 1.75$ policy gives lower soft penalties.
In more details, the real D-Wave processing is tied to some parameters both of the particular QUBO and the machine itself. We have achieved best results for the coupling constant $c_{ss} = 2.0$ for the small example (a) the same observation we have found for the large example. This was not expected as the coupling between quantum bits representing single classical bit was rather week there. Here we probably take advantage from some possible variations within the realization of a logical bit. This observation demonstrates that the embedding selection may be meaningful in searching for the convergence towards proper solutions lying in the low energy part of the spectrum. For the small cases we have observed the feasible solution for relative small number of samples equal to 1k. For the larger case the number of samples had to be increased to the maximal possible equal to 250k and still we have not reached any feasible solution. The conclusion here is that the impact of the noise amplifies a lot with the size of the problem. The convergence of the best achieved solution toward the optimal one with the sample size is complex and the deep statistical analysis concerning sampling the real Boltzman distribution of the annealer is required there.

As demonstrated in Fig. 9(b) for some cases only one hard constraint was broken. This may point out that we are near the region of feasible solutions. However, soft constraints’ penalty values there are still far from optimal achieved by means of simulations, see Tab. 3. To elucidate the interplay between penalties we refer to Fig. 10, where the solutions are not feasible but can be easily corrected by the dispatcher to have feasible ones. In 10(b) the corrected solution would be optimal while in 10(a) it would not and other than all other achieved solutions. Hence the use of the current quantum annealer would rather sample the more excited part of the QUBO spectrum and can sometimes provide the non-usual solution. Such solution however can be applied by the dispatcher for some particular reasons not encoded directly in the QUBO. These reasons may concern: temporal dispatching problems, rolling stock emergency, non standard requirements etc.

6 Discussion on results and conclusions

We have introduced a new approach to the single-track line dispatching problem that can be implemented on a real quantum annealing device (D-Wave 2000Q). We have solved our model on the quantum annealer and also using classical algorithms capable of solving smaller instances of these models.

In particular, we have addressed two particular real-life railway dispatching problems from the practice in Poland; many similar examples exist in other networks, too. We have introduced a quadratic unconstrained binary optimization (QUBO) model of the problem which can be solved with quantum annealing.

The first dispatching problem we considered (Nidzica – Olsztynek section of the line No. 216) was fairly small and is thus defined using “only” 48 logical quantum bits (which we were able to embed into 373 physical quantum bits of the real quantum processor). The final state reached by the quantum annealer for this problem turned out to be optimal for many parameter settings. This highlights that small-sized dispatching problems are already within the reach of near-term quantum annealers.

Our second dispatching problem (Goleszów – Wisła Uzdrowisko section of the line No. 191) was larger and
needed 198 logical quantum bits. Here, the number of physical quantum bits depends on the number of constraints in each of the analyzed cases. We were able to embed all of our 4 dispatching cases on the 191 railway line on the ideal Chimera graph (2048 physical quantum bits) using a state-of-the-art embedding algorithm. This could be solved via a classical calculation of the annealing result. Meanwhile, on the physical device (whose graph is not perfect and lacks several quantum bits and couplings), we were able to embed only three out of four of our cases (except for case 3, where the number of conflicts is the highest). We expect that such obstacles will disappear soon as new embedding algorithms are being developed for both current Chimera topology and the incoming D-Wave Pegasus 5000Q one, see for example [58, 59]. Therefore, it is not unreasonable to expect that the range of problems that are embeddable so that they can be solved on physical hardware will substantially increase in near future. Unfortunately, the D-Wave 2000Q solutions of our second problem appeared to be non-optimal and not feasible for dispatchers. This is attributable to the noise still present in the current quantum machine. As the quantum annealing is an emerging technology, the upcoming generation of quantum annealers are expected be more and more accurate, and less prone to errors.

We are aware that the examples of the single track railway dispatching problem discussed in the paper can be considered as trivial from the point of view of professional dispatchers. Our intention, however, was to provide a proof-of-concept demonstration of the applicability of quantum annealing in this problem. Due to the tight size of the current quantum annealing processors, our implementation is limited. This is about to change soon as new quantum annealing processors are expected – i.e., D-Wave Pegasus 5000Q. Moreover, these processors promise unprecedented speed (time to solution ranges from microsecond to milliseconds).

Meanwhile we have successfully solved our model using certain algorithms running on classical computers; CPLEX, and the novel Tensor Network method. Therefore our approach offers an alternative approach to the problem: QUBO modeling and the use of quantum-motivated classical algorithms. While these obviously do not offer a breakthrough in the scaleability, they are essential for the validation and assessment of the results of real quantum annealing. In addition, they can yield practically useful results.

With the development of the quantum annealers, the problem sizes they are about to handle will be bigger and bigger. At some point, they will cross the point in which neither manual approaches nor conventional algorithms would be capable to solve them in a reasonable time, what is a subject of many complex dispatching situations for which no optimal solutions are possible at present. Then, the quantum technology or quantum-inspired algorithms raise the prospect to tackle such problems successfully. This gives the incentive to keep up with the development of the quantum devices and related algorithms for railway applications.

7 Acknowledgments

This work was supported by the Foundation for Polish Science (FNP) under grant number TEAM NET POIR.04.04.00-00-17C1/18-00 (KD and BG) and the National Science Centre (NCN), Poland, under the project number 2016/22/E/ST6/00062 (KJ), and the National Research, Development and Innovation Office of Hungary, under project numbers K133882 and No. K124351, and No. 2017-1.2.1-NKP-2017-00001 HunQuTech.

References

[1] N. Dattani, S. Szalay, and N. Chancellor, “Pegasus: The second connectivity graph for large-scale quantum annealing hardware.” 2019.
[2] P. A. R. Ade et al., “Planck 2015 results - XIII. Cosmological parameters,” Astron. Astrophys., vol. 594, p. A13, 2016.
[3] K. Jałowiecki, M. M. Rams, and B. Gardas, “Brute-Force approach to spin-glass problems with CUDA.” 2019.
[4] T. Lanting and others, “Entanglement in a Quantum Annealing Processor,” Phys. Rev. X, vol. 4, p. 021041, May 2014.
[5] J. Törnquist and J. A. Persson, “N-tracked railway traffic re-scheduling during disturbances,” Transportation Research Part B: Methodological, vol. 41, no. 3, pp. 342 – 362, 2007.
[6] L. Lamorgese, C. Mannino, D. Pacciarelli, and J. T. Krassmann, pp. 265–283. Cham: Springer International Publishing, 2018.
[7] J. Jensen, O. Nielsen, and C. Prato, “Passenger perspectives in railway timetabling: A literature review,” Transport Reviews, vol. 36, no. 4, pp. 500–526, 2016.
[8] C. Wen, P. Huang, Z. Li, J. Lessan, L. Fu, C. Jiang, and X. Xu, “Train dispatching management with data-driven approaches: A comprehensive review and appraisal,” IEEE Access, vol. 7, pp. 114547–114571, 2019.
[9] X. Cai and C. J. Goh, “A fast heuristic for the train scheduling problem,” Computers & Operations Research, vol. 21, no. 5, pp. 499 – 510, 1994.
[10] B. Szpigel, “Optimal train scheduling on a single line railway,” Operational Research, vol. 72, pp. 343–352, 1973.
[11] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd ed., 2008.
[12] J.-F. Cordeau, P. Toth, and D. Vigo, “A survey of optimization models for train routing and scheduling,” *Transportation Science*, vol. 32, p. 380–404, Apr. 1998.
[13] J. Törnquist, “Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms,” in *5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS’05)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.
[14] T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel, *Delay Propagation and Delay Management in Transportation Networks*, pp. 285–317. Cham: Springer International Publishing, 2018.
[15] F. Corman and L. Meng, “A review of online dynamic models and algorithms for railway traffic management,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1274–1284, June 2015.
[16] V. Cacchiani and P. Toth, “Nominal and robust train timetabling problems,” *European Journal of Operational Research*, vol. 219, no. 3, pp. 727–737, 2012. Feature Clusters.
[17] I. Hansen, *State-of-the-art of railway operations research*, pp. 35–47. United Kingdom: WIT Press, 2010. Zie virtueelboek; Verbeterde versie van een paper dat ingediend is voor Comprail X in 2006.
[18] J. Lange and F. Werner, “Approaches to modeling train scheduling problems as job-shop problems with blocking constraints,” *J Sched.*, vol. 21, pp. 191–207, Apr. 2018.
[19] A. Mascis and D. Pacciarelli, “Job-shop scheduling with blocking and no-wait constraints,” *European Journal of Operational Research*, vol. 143, no. 3, pp. 498–517, 2002.
[20] A. D’Ariano, D. Pacciarelli, and M. Pranzo, “A branch and bound algorithm for scheduling trains in a railway network,” *European Journal of Operational Research*, vol. 183, no. 2, pp. 643–657, 2007.
[21] D. Venturelli, D. J. Marchand, and G. Rojo, “Job-shop scheduling solver based on quantum annealing,” *aist S ing an*, p. 25, 2016.
[22] X. Zhou and M. Zhong, “Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds,” *Transportation Research Part B: Methodological*, vol. 41, pp. 320–341, Mar. 2007.
[23] S. Harrod, “Modeling Network Transition Constraints with Hypergraphs,” *Transportation Science*, vol. 45, pp. 81–97, Feb. 2011.
[24] L. Meng and X. Zhou, “Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables,” *Transportation Research Part B: Methodological*, vol. 67, pp. 208–234, Sept. 2014.
[25] Y. Jing, Y. Liu, and M. Bi, “Quantum-inspired immune clonal algorithm for railway empty cars optimization based on revenue management and time efficiency,” *Cluster Computing*, vol. 22, pp. 545–554, 2019.
[26] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge, UK: Cambridge University Press, 2010.
[27] A. Lucas, “Ising formulations of many NP problems,” *Front. Phys.*, vol. 2, p. 5, 2014.
[28] T. Albash and D. A. Lidar, “Adiabatic quantum computation,” *Rev. Mod. Phys.*, vol. 90, p. 015002, Jan 2018.
[29] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, “Physics-inspired optimization for quadratic unconstrained problems using a digital annealer,” *Front. Phys.*, vol. 7, p. 48, 2019.
[30] D. Pierangeli, M. Rafayelyan, C. Conti, and S. Gigan, “Scalable spin-glass optical simulator.” 2020.
[31] Y. Yamamoto, K. Alhara, T. Leleu, K. Kawarabayashi, S. Kako, M. Fejer, K. Inoue, and H. Takesue, “Coherent Ising machines—optical neural networks operating at the quantum limit,” *Npj Quantum Inf.*, vol. 3, no. 1, p. 49, 2017.
[32] B. H. Fukushima-Kimura, S. Handa, K. Kamakura, Y. Kamijima, and A. Sakai, “Mixing time and simulated annealing for the stochastic cellular automata.” 2020.
[33] F. Cai et al., “Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks,” *Nat. Electron.*, 2020.
[34] J. Avron and A. Elgart, “Adiabatic theorem without a gap condition,” *Commun. Math. Phys.*, vol. 203, pp. 445–463, Jun 1999.
[35] S. Sachdev, *Quantum Phase Transitions*. Cambridge University Press, 2011.
[36] J. Dziarmaga, “Dynamics of a Quantum Phase Transition: Exact Solution of the Quantum Ising Model,” *Phys. Rev. Lett.*, vol. 95, p. 245701, Dec 2005.
[37] J. Dziarmaga, “Dynamics of a quantum phase transition and relaxation to a steady state,” *Adv. Phys.*, vol. 59, no. 6, pp. 1063–1189, 2010.
In the appendix we present all the solutions of the dispatching problems on the line No. 191 obtained by our algorithms. Both the CPLEX and tensor network approaches (which are based on QUBO) allow for rather arbitrary decisions on train prioritization. These approaches focus on the train delay propagation on subsequent trains, as illustrated by the comparison of all the solutions of case 1. Provided the Ks2 is delayed, additional delay of Ks3 would happen (what we call a “cascade effect”). Furthermore the tensor network output in Fig. 16 demonstrates the degeneracy of the ground state and the solutions in the low excited state, which however, does not have a relevant impact on the dispatching situation.

Note that the simple heuristics (FCFS, FLFS) sometimes return trouble-causing solutions. These suggest a solution where one train needs to have a time-consuming stopover on the particular station, see Figs. 12(c), 13(b). (Such problems sometimes appear in the real life train dispatching too.) Finally, if the problem is easily solvable as in case 4, all the methods analyzed in the paper give the same solution. This serves as a quality test of our method.
(a) case 1 – simple conflict, observe that the additional delay of Ks2 will propagate to the delay of Ks3.

(b) case 2 – two conflicts, similar to 11(a), but with no impact of Ks2 on Ks3.

(c) case 3 – multiple conflict.

(d) case 4 – conflict straightforward to resolve.

Figure 11: The conflicted timetables, various types of conflicts.
(a) case 1 – “cascade effect”, the delay of Ks2 causes further delay of Ks3.

(b) case 2 – optimal solution reached rather “at random”: probably it is reached just because of the relative simplicity of the problem.

(c) case 3 – a problematic solution with undesirably long waiting times of certain trains, observe the stopover of Ks2.

(d) case 4 – optimal solution according to all methods

Figure 12: The FCFS solutions, some with a trouble causing stopover of the particular train.
(a) case 1 – optimal solution reached “at random”, as its duplication in (b) case 2 – duplication of the solution in Fig. 13(a) causing an stopover of Ks3. (c) case 3  
(d) case 4 – optimal solution according to all methods

Figure 13: The FLFS solutions, some with a trouble causing stopover of the particular train.
Figure 14: The AMCC solutions. The minimization of the maximal secondary delays from AMCC excludes the inacceptably long stopovers as in Figs. 12(c), 13(b). However these solutions do not exclude the propagation of smaller delays amongst several trains ("cascade effect") see Fig. 14(a).
Figure 15: The CPLEX solutions: exact ground states of the QUBOs. There are no inacceptably long stopovers. Further, the trains’ prioritization and the delay propagation to subsequent trains is taken into account.
Figure 16: The tensor network solutions, although not always the exact ground states were achieved, the solutions are equivalent from the dispatching point of view with those in Fig. 15.
This figure "line.png" is available in "png" format from:

http://arxiv.org/ps/2010.08227v1
This figure "line_small.png" is available in "png" format from:

http://arxiv.org/ps/2010.08227v1