Railway Timetable Optimization

Inaugural Dissertation
Presented to the
School of Business and Economics
Freie Universität Berlin
in Candidacy for the Degree of
doctor rerum politicarum (Dr. rer. pol.)

Julian Reisch, M. Sc.

Berlin, 2021

Referees:

1. Univ.-Prof. Dr. Natalia Kliewer, Freie Universität Berlin
2. Univ.-Prof. Dr. Ralf Borndörfer, Freie Universität Berlin
3. Prof. Dr.-Ing. Johannes Schlaich, Beuth Hochschule Berlin

Date of Disputation:

27.04.2021

Publications:

- Reisch, J., Kliewer, N., Martin, U., Pöhle, D. (2021). "Bestimmung der Kapazitätssteigerung durch Einführung der Mittelpufferkupplung und ep-Bremse". In: *ETR - Eisenbahntechnische Rundschau 1-2/2021.*,
URL: https://www.eurailpress.de/publikationen/etr.html

- Reisch, J., Großmann, P., Pöhle, D., Kliewer, N. (2021). "Conflict Resolving - A Local Search Algorithm for Solving Large Scale Conflict Graphs in Freight Railway Timetabling". In: *European Journal of Operational Research.* DOI: 10.1016/j.ejor.2021.01.006

- Reisch, J. (2020). "State of the Art Overview on Automatic Railway Timetable Generation and Optimization". In: *Diskussionsbeiträge Fachbereich Wirtschaftswissenschaft 2020/20.* DOI: 10.17169/refubium-28658

- Reisch, J., Großmann, P., Kliewer, N. (2020). "Stable Resolving - A Randomized Local Search Heuristic for MaxSAT". In: Schmid U., Klügl F., Wolter D. (eds) *KI 2020: Advances in Artificial Intelligence*. KI 2020. Lecture Notes in Computer Science, vol. 12325, pp. 163–175, Springer, Cham.
DOI: 10.1007/978-3-030-58285-2_12

- Lindner, N., Reisch, J. (2020). "Parameterized Complexity of Periodic Timetabling". Preprint published in: ZIB-Report (20-15). URN: urn:nbn:de:0297-zib-78314. Under revision for the *Journal of Scheduling*.

- Reisch, J., Kliewer, N. (2020). "Black-Box Optimization in Railway Simulations". In: Neufeld J.S., Buscher U., Lasch R., Möst D., Schönberger J. (eds) *Operations Research Proceedings 2019*. Operations Research Proceedings (GOR (Gesellschaft für Operations Research e.V.)), pp. 717–723, Springer, Cham. DOI: 10.1007/978-3-030-48439-2_87

- Hauck, F., Kliewer, N., Reisch, J., Rößler, D. (2020): "Datengetriebene Feinjustierung von Fahrplänen zur Erhöhung der Pünktlichkeit im Schienenverkehr". In: Tagungsbericht Heureka 2021, FGSV Verlag, Köln. (Not part of this dissertation).

- Reisch, J., Großmann, P., Kliewer, N. (2019). "Conflict Resolving - A Maximum Independent Set Heuristics for Solving MaxSAT". In: Andrej Brodnik, Gábor Galambos, Branko Kavšek (eds) *Proceedings of the 22nd International Multiconference Information Society*, vol. 1, pp. 67–71.
URL: http://library.ijs.si/Stacks/Proceedings/InformationSociety/2019/IS2019_Volume_I%20-%20MATCOS.pdf

- Rößler, D., Reisch, J., Kliewer, N. (2019). "Modeling Delay Propagation and Transmission in Railway Networks". In: Thomas Ludwig and Volkmar Pipek (eds) *Human Practice. Digital Ecologies. Our Future. 14. Internationale Tagung Wirtschaftsinformatik (WI 2019), February 24-27, 2019, Siegen, Germany*, vol. 14, pp. 98–111, University of Siegen, Germany / AISeL.
URL: https://aisel.aisnet.org/wi2019/track02/papers/2/

Declaration of Authorship

Except where reference is made in the text, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis presented by me for another degree or diploma. No other person's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any other degree or diploma in any other tertiary institution.

Abstract

In this cumulative dissertation, we study several aspects of railway timetable optimization. The first contributions cover *Practical Applications of Automatic Railway Timetabling*. In particular, for the problem of simultaneously scheduling all freight trains in Germany such that there are no conflicts between them, we propose a novel column generation approach. Each train can choose from an iteratively growing set of possible routes and times, so called slots. For the task of choosing maximally many slots without conflicts, we present and apply the heuristic algorithm Conflict Resolving (CR). With these two methods, we are able to schedule more than 5000 trains simultaneously, exceeding the scopes of other studies. A second practical application that we study is measuring the capacity increase in the railway network when equipping freight trains with electro-pneumatic brakes and middle buffer couplings. Methodically, we propose to explicitly construct as many slots as possible for such trains and measure the capacity as the number of constructed slots. Furthermore, we contribute to the field of *Algorithms and Computability in Timetable Generation*. We present two heuristic solution algorithms for the Maximum Satisfiability Problem (MaxSAT). In the literature, it has been proposed to encode different NP-complete problems that occur in railway timetabling in MaxSAT. In numerical experiments, we prove that our algorithms are competitive to state-of-the-art MaxSAT solvers. Moreover, we study the parameterized complexity status of periodic scheduling and give proofs that the problem is NP-complete for input graphs of bounded treewidth, branchwidth and carvingwidth. Finally, we propose a framework for analyzing *Delay Propagation in Railway Networks*. More precisely, we develop delay transmission rules based on different correlation measures that can be derived from historical operations data. What is more, we apply SHAP values from Explainable AI to the problem of discerning primary delays that occur stochastically in the operations, to secondary follow-up delays. Transmission rules that are derived from the secondary delays indicate where timetable adjustments are needed. In our last contribution in this field, we apply such adjustment rules for black-box optimization of timetables in a simulation environment.

Zusammenfassung

Diese kumulative Dissertation befasst sich mit verschiedenen Aspekten der Fahrplanoptimierung im Bahnverkehr. Sie umfasst erstens das Themenfeld *praktischer Anwendungen automatischer Fahrplanung*. Hier wird ein Spaltengenerieungsansatz vorgestellt, um simultan alle Güterzüge in ganz Deutschland zu planen, ohne dass Konflikte entstehen. Dabei wird für jeden Zug eine Trasse aus einer Menge möglicher Trassen ausgewählt. Es wird ein heuristischer Algorithmus vorgestellt, der für maximal viele Züge je eine Trassen auswählt, sodass je zwei ausgewählte Trassen keinen Konflikt zueinander haben. Dieser Ansatz ermöglicht es, mehr als 5000 Güterzüge gleichzeitig zu planen. Dies übersteigt den Umfang bisheriger Ansätze. Eine zweite praktische Anwendung automatischer Fahrplanung ist die Messung der Kapazitätssteigerung, die sich ergibt, wenn Güterzüge mit elektropneumatischer Bremse und Mittelpufferkupplung ausgestattet werden. Die Kapazität wird hierbei gemessen in der Anzahl von Trassen, die sich automatisch konstruieren lassen, ohne dass Konflikte entstehen. Des Weiteren befasst sich diese Arbeit mit dem Themenfeld *Algorithmen und Berechenbarkeit in der Fahrplanung*. Dabei werden zwei heuristische Algorithmen zur Lösung des Maximum Satisfiability (MaxSAT) Problems präsentiert. Verschiedene NP-vollständige Probleme aus der Fahrplanung lassen sich als MaxSAT Problem kodieren. In numerischen Experimenten weisen wir nach, dass diese beiden Algorithmen vergleichbare Ergebnisse erzielen wie state-of-the-art Algorithmen aus der Literatur. Außerdem wird der Komplexitätsstatus des periodischen Fahrplanungsproblems PESP untersucht. Es wird bewiesen, dass das PESP NP-vollständig ist auch auf Graphen beschränkter Baumweite, Verzweigungsweite und Schnittweite. Im letzten Themenfeld *Verspätungsfortpflanzung in Bahnnetzen* werden Verspätungsübertragungsregeln aus historischen Verspätungsdaten abgeleitet und anhand von Korrelationskoeffizienten beschrieben. Ferner werden SHAP values angewandt, um Primär- von Sekundärverspätungen zu unterscheiden. Diejenigen Übertragungsregeln, die aus Sekundärverspätungen abgeleitet werden, dienen als Hinweise, wo Fahrplananpassungen zu weniger Verspätungsübertragung und mehr Pünktlichkeit führen können. Diese Anpassungen werden angewandt und durch Simulationen evaluiert.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Railway timetable generation is the task of assigning a slot to a train trip. A slot is a technically valid path through the network from the trains origin to its destination together with the times the train passes the signals on this path. In most countries, a train is only allowed to operate when it has been assigned a slot (Hansen and Pachl, 2014). In this respect, railway transportation is different to other means of transportation. For instance, driving a car on a road offers much more flexibility as the route can be calculated and altered during the ride. The main reason why train trips are planned beforehand and on a detailed level is safety. Trains usually have very long brake paths and moreover, changing switches for trains takes time. Therefore, a railway undertaker who wants to operate a trains first has to address a train path request to the infrastructure manager specifying the train characteristics (e.g. which locomotive, how many coaches), the origin and destination of the trip and the desired departure (or arrival) time. The infrastructure manager then constructs a slot for each requested train trip. In this process of constructing and coordinating possibly many requests, the infrastructure manager has different and sometimes conflicting aims. Firstly, the slot is the product of the infrastructure manager that one wants to sell. Therefore, the slot should meet the request as well as possible and with a minimal travelling time. Secondly, the infrastructure manager wants to sell slots to as many railway undertakers as possible. Hence, the capacity utilization of its most valuable resource, the infrastructure, should be maximized. Thirdly, the constructed timetable should be possible to operate. That is, given that there are stochastically occurring disturbances in the operations, the timetable should have enough time supplements and buffer times to compensate delays and prevent follow-up delays. With these three objectives, railway timetabling becomes an optimization problem.

1.2. Thesis Outline

In this cumulative dissertation, several aspects of automatic railway timetable generation and optimization are considered. There are 9 contributions grouped in 3 *dimensions* as shown in Table 1.1. With the paper by Reisch (2020), we start with an overview on state-of-the-art techniques for automatic railway timetable generation

Table 1.1.: Mapping of Research Publications to Chapters and Dimensions

Chapter	Title	Dimension
2	State of the Art Overview on Automatic Railway Timetable Generation and Optimization	-
3	Conflict Resolving - A Local Search Algorithm for Solving Large Scale Conflict Graphs in Freight Railway Timetabling	(1)
4	Bestimmung der Kapazitätssteigerung durch Einführung der Mittelpufferkupplung und ep-Bremse	(1)
5	Conflict Resolving - A Maximum Independent Set Heuristics for Solving MaxSAT	(2)
6	Stable Resolving - A Randomized Local Search Heuristic for MaxSAT	(2)
7	Parameterized Complexity of Periodic Timetabling	(2)
8	Modeling Delay Propagation and Transmission in Railway Networks	(3)
9	Discerning Primary and Secondary Delays in Railway Networks using Explainable AI	(3)
10	Black-Box Optimization in Railway Simulations	(3)

and optimization[1]. We present the challenges in this field of research and review the literature with respect to models, solution approaches and applications in practice. In particular, we discuss algorithms to automatically construct a slot for a train. Pöhle (2016) has proposed that the German infrastructure manager *DB Netze* today's manual slot construction shall be enhanced by automatization for calculating better slots in shorter time. Furthermore, we present the differences between periodic and aperiodic timetabling and discuss the respective complexity status. It will turn out that the aperiodic problem can be solved using a polynomial time shortest path algorithm while the periodic problem, modelled as the Periodic Event Scheduling Problem (PESP) is proved to be NP-complete (M. Odijk, 1994). What is more, we study the problem of planning multiple trains simultaneously, denoted by the Train Path Assignment Problem (TPAP). This problem can be modelled as a multi-commodity flow and is hence NP-complete, as well (Even et al., 1975). Finally, we present models for robust timetabling that aims at minimizing the sum of expected follow-up delays when minor disturbances occur. The most common approaches either incorporate robustness in the timetable generation itself, or modify an existing timetable with the aim of improving its robustness.

The remaining papers are mapped to one of the following three dimensions. Dimension (1) covers **Practical Applications of Automatic Railway Timetabling**. In the first paper (Reisch, Großmann, Pöhle, et al., 2021), we study the TPAP - the problem

[1]The paper (Reisch, 2020) was created in sole authorship. The other publications were created in co-authorship. All authors contributed equally to the papers.

of scheduling multiple trains simultaneously - for all freight trains in Germany with real-world data provided by DB Netze. Due to the large scale of the application, we propose a novel column generation approach for the multi-commodity flow modelling of the TPAP. In each iteration, we choose one slot per train from a growing set of possible slots such that no two chosen slots have a conflict (Zwaneveld et al., 1996). This task can be modelled in a graph where two vertices (slots) share an edge if the respective slots have a conflict. A maximum set of pairwise conflict-free slots can be found with an independent set algorithm. We present *Conflict Resolving (CR)* which is a local search independent set heuristic that is tailored to the specially structures graphs from this application. With the column generation together with the independent set heuristic, we are able to schedule more than 5000 freight trains simultaneously, extending the scopes of other approaches significantly. For constructing the single slots in this framework, we employ a heuristic slot construction algorithm by Dahms et al. (2019). This algorithm is also in use in the second paper of the first dimension (Reisch, Kliewer, et al., 2021). Here, we use the slot construction algorithm to quantify the capacity of track sections. More precisely, we construct as many slots as possible on the track section for a representative set of train trips. Then, the capacity of the section is the number of constructed slots. To our knowledge, this approach for measuring capacities in railway networks has not been studied before in the literature. We apply this method for the case study of measuring the capacity increase when equipping freight trains with electro-pneumatic brakes and middle buffer couplings as proposed by Martin et al. (2015). It turns out that this upgrade increases the freight railway capacity in Germany by 4%.

The second dimension (2) **Algorithms and Computability in Timetable Generation** studies more theoretic aspects of railway timetabling. The first two papers (Reisch, Großmann, and Kliewer, 2019; Reisch, Großmann, et al., 2020) address solution algorithms for the (weighted) Maximum Satisfiability Problem (MaxSAT). In the Satisfiability problem (SAT), we are given a set of Boolean variables together with signs that are grouped in clauses. Within the clauses, the variables are joint by conjunctions and the clauses are joint by disjunctions. For example, if $x \vee y$ and $\neg x \vee y \vee z$ are two clauses, the entire formula will be $(x \vee y) \wedge (\neg x \vee y \vee z)$. Then, the task is to assign either *true* or *false* to each variable such that all clauses are satisfied. (Weighted) MaxSAT is the incomplete version of this task where one aims at finding an assignment satisfying a (weighted) maximum subset of clauses. Many problems from combinatorial optimization can be encoded as MaxSAT. In particular, in the book by Bacchus, Berg, et al. (2020) we describe the MaxSAT encoding of the Independent Set problem for the TPAP graph instances as proposed by Reisch, Großmann, Pöhle, et al. (2021). Furthermore, the PESP can be encoded as MaxSAT, as well (Großmann et al., 2012a). As the formulas in the railway timetabling application are of large scale, we propose two heuristic algorithms. The first one works by transforming the MaxSAT instance to an instance of the Independent Set problem and solving it using CR (Reisch, Großmann, and Kliewer, 2019). The drawback of this approach is that the transformation multiplies the sizes of the instances so that the solvers performance often is uncompetitive. Hence, we

developed *Stable Resolving (SR)* which also is a local search heuristic that - unlike CR - directly works on the MaxSAT formulas (Reisch, Großmann, et al., 2020). A comparison to other MaxSAT solvers from the 2019 MaxSAT challenge[2] shows that SR now yields comparable results, in particular on the railway transport application. The last contribution in this dimension investigates the parameterized complexity status of the PESP (Lindner and Reisch, 2020). That is, if we restrict the input graph to belong to a particular graph class, does the problem remain NP-complete or is there a polynomial time algorithm that makes use of the special graph types? We answer this question for the three classes of graphs with treewidth at least 2, branchwidth of at least 2 and carvingwidth of at least 3. In each case, the PESP remains NP-complete. Nevertheless, we give two pseudo-polynomial-time dynamic programming algorithms solving the PESP on input graphs with bounded tree- or branchwidth.

In the third dimension (3) **Delay Propagation in Railway Networks** we analyze the emergence of follow-up delays and how a timetable can be optimized so that the follow-up delays are as small as possible. In railway operations, there are stochastically occurring disturbances and the timetable should have enough time supplement to compensate them. In addition, to avoid that the delay is propagated to a following train, there should be sufficient buffer times between the train trips. In (Rößler et al., 2019) we derive correlation rules from historical railway operations data provided by DB Netze on how the delays of different trains affect each other. For example, if the data reveal that one train almost every day waits for passenger of a second train, then we derive a transmission rule between the two trains. Such rules can then be used for optimally allocating buffer times. However, we do not differentiate between primary and secondary delays in this paper. Primary delays are the type of delays that occur even if the train is the only train in the network. Vehicle or infrastructure malfunctions are typical examples of primary delay causes. Secondary delays are the resulting follow-up delays from one train to another. For the transmission rules, we are only interested in secondary delays. Hence, we propose to apply Explainable AI for discerning primary from secondary delays in the data (Rößler et al., 2021). More precisely, we train a machine learning model on primary and secondary features for predicting the total delay, that is, the sum of primary and secondary delay. Then, we compute SHAP values (Lundberg and Lee, 2017) for extracting the secondary features' contribution to the total amount of predicted delay. With this approach, we can discern the secondary part of the total delays. The last paper in this dimension presents a Markov-chain simulation framework for delay propagation (Reisch and Kliewer, 2020). Given a timetable and historical delay data, this framework allows for adjusting the time supplements and then evaluates the achieved decrease of expected delays by running the simulation. The adjustments can be derived from the data as proposed in the first two papers in this dimension, or can be based on expert knowledge.

[2]MaxSAT Evaluation 2019 https://maxsat-evaluations.github.io/2019/index.html

2. State of the Art Overview on Automatic Railway Timetable Generation and Optimization

Abstract

In railway transportation, each train needs to have a timetable that specifies which track at which time will be occupied by it. This task can be addressed by automatization techniques both in generating a timetable and in optimizing an existing one. In this paper, we give an overview on the state of the art of these techniques. We study the computation of a technically valid slot for a train that guarantees a (short) spatial and temporal way through the network. Furthermore, the construction of a cyclic timetable where trains operate e.g. every 60 minutes, and the simultaneous construction of timetables for multiple trains are considered in this paper. Finally, timetables also need to be robust against minor delays. We will review the state of the art in the literature for these aspects of railway timetabling with respect to models, solution algorithms, complexity results and applications in practice.

Contents

2.1. Introduction

Before a train can operate, it is mandatory that it has a timetable. Therefore, railway undertakers request timetables at the infrastructure manager for their train operations. The task of the infrastructure manager is to coordinate these requests and create a timetable for all operating trains. In this paper, we review the literature on the state of the art on how this coordination and generation process can be enhanced by automatization. That is, for the different planing horizons, periodicities and objectives, we present mathematical models, solution algorithms, complexity results and application tools.

Single Slot Construction. The timetable of a single train is called a slot. Most of the European infrastructure managers use the blocking time model to construct a slot (Hansen and Pachl, 2014). In the blocking time model, a slot is a sequence of block segments. Each block segment is defined from one signal to the next one and has a temporal expansion that reflects the trains driving dynamics. If two slots of different trains have overlapping block segments, we say that the trains have a conflict.

Planning Horizons. Depending on the planning horizon, the infrastructure manager has different degrees of flexibility to schedule the requested trains. In the annual timetable, all train operation requests, both for passenger and freight trains, are collected and then considered simultaneously. The request to operate a train comes with a desired planning period, for example, only on weekdays. Due to requests from other trains, or due to infrastructure restrictions, the infrastructure manager can split the planning period and construct for each part a separate slot. No two trains can have conflicting slots. For shorter planning horizons and especially in the intraday ad-hoc timetabling, there already exist the timetables from the annual timetable and the new slots need to be constructed individually and without any conflicts to existing slots.

Periodicities. Especially for passenger trains, the railway undertakers often wish to operate their train periodically, that is, for example, every 60 minutes. Periodical timetabling is mostly relevant for the annual timetable and ad-hoc train operations run aperiodically.

Objectives. As most infrastructure managers construct their timetables manually, one objective in automatic timetable generation simply is to generate a timetable without any manual work but by automatized algorithms. Moreover, in the models for timetable generation, typically three objectives are considered and balanced out against each other. First, the capacity utilization shall be maximized, that is, as many trains as possible should operate. Second, the timetable quality shall be maximized which means that the slots should have minimal running times. Third, the timetables robustness shall be maximized. That is, the expected follow-up delays shall be minimized given minor disturbances.

Generation versus Adaption. Timetable optimization can take place in the generation of the timetable, or in adapting an existing timetable to improve some objective function.

The outline of this paper is as follows. In Section 2.2, we review models to construct single slots in a time-space network that is restricted by capacity utilization. This task comes in ad-hoc planning, for example, when other trains already use some parts of the infrastructure. Then, in Section 7.2, we point out the difference between aperiodic and periodic timetabling. In particular, we present the Periodic Event Scheduling Problem (PESP) and analyze its complexity. Another complexity in timetabling is planning multiple trains simultaneously, as needs to be done in the annual timetable. Section 2.4 is about modelling and solving this problem, which is called the Train Path Assignment Problem (TPAP). In Section 2.5, we review timetabling optimization models, for both generation and adaption. Finally, in Section 2.6, we conclude this paper.

2.2. Slot Construction

A slot $y = ((x,t)_i)_{i=0}^n$ of a train is a trajectory through time and space. It specifies the time t a position x (mostly a signal) is passed. The time difference $t' - t$ between x and x' depends on the infrastructure and the train characteristics. Mostly, the starting time interval, the starting position and the target position are given. Then, in this simplest model, a slot can be computed as a constrained path through the time-discretized time-expanded infrastructure graph (cf. e.g. A. Caprara et al. (2002), Zhang et al. (2019)). For a mathematical formulation **MIP-***slot*, let $\mathbf{F}$ denote its incidence matrix. That is, $\mathbf{F}$ is -1 for each arc entering a particular position x at a particular time t and $+1$ for each arc leaving x at t. The set of arcs that can be reached from an arc depends on the train characteristics, such as maximum speed, and the infrastructure graph. Furthermore, the indicator vector I is 1 for each arc in the starting position in the starting interval in time, -1 for the destination arcs, and zero elsewhere. Thus, (1b) ensures flow conservation. The conflict matrix $\mathbf{C}$ ensures that the slot y can only occupy tracks at times that are not occupied, yet. Finally, (1d) indicates which arcs the slot uses. The objective (1a) minimizes the total travel time of the slot.

$$\begin{aligned}
\textbf{MIP-}slot \text{ Minimize} \quad & t_n - t_0 && && \text{(2.1a)}\\
\text{s.t.} \quad & \mathbf{F}y = I && && \text{(2.1b)}\\
& \mathbf{C}y \leq \mathbf{1} && && \text{(2.1c)}\\
& y_i \in \{0,1\} && \text{for each } y_i = (x,t)_i && \text{(2.1d)}
\end{aligned}$$

In particular, the slot construction **MIP-***slot* is polynomially solvable when applying a shortest path or flow algorithm (Ford and Fulkerson, 1956). However, modelling the driving on a track only by the time span the train takes to pass the track is too rough for most infrastructure managers. A more sophisticated model describes a slot as a sequence of block segments as defined in Hansen and Pachl (2014). A block segment

is the time span a track is utilized by a train in which no other train can enter the track including overlaps and headway times and depending on the concrete driving dynamics of a train. For example, if a train halts, depending on its load, it might block the succeeding track for some time because there might be an overlap. Figure 2.1 illustrates the construction of a slot as a sequence of block segments in the capacity that remains after other slots have been planned already. The German infrastructure manager DB Netz works with the block segment model. Therefore, Dahms et al. (2019) propose a heuristic shortest path algorithm to compute a slot with its block segments automatically. In the application *click and ride*, this algorithm comes into practice for the construction of ad-hoc slots for freight trains.

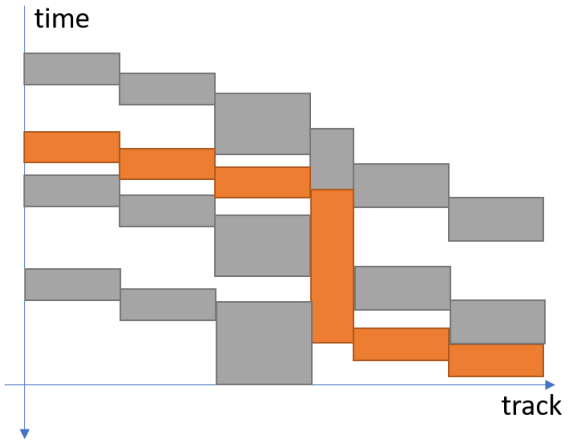


Figure 2.1.: Block segments in the time-track diagram: The orange train can be scheduled in the capacity that remains after three other trains (grey) have been scheduled.

2.3. (A-)Periodic Timetabling

In this section, we will discuss the additional task of a timetable to satisfy periodicity constraints. That is, a train is supposed to be operated every T minutes with the same timetable. We denote T by the period. Following Serafini and Ukovich, 1989a, the periodic timetabling problem is modelled in the event network.

Definition 2.3.1. An *event network* is a tuple $\mathcal{N} = (D, l, u)$ where $D = (V, A)$ is a directed graph and $l, u \in \mathbb{Q}^{|A|}$ such that $l_a \leq u_a$ for all $a \in A$. The vertices V and arcs A are called events and activities, respectively.

Events are e.g. passings of a train at a given station or signal. Activities are the driving from one signal to the next one or the headway times from one train to a succeeding train. Hence, multiple trains can be scheduled simultaneously. Unlike in the time-discretized time-expanded infrastructure graph, the events are fixed. That is,

deviations in the spatial way a train takes or different overtaking opportunities are not considered. The only flexibility is the time differences between the events.

Definition 2.3.2. Given an event network $\mathcal{N} = (D, l, u)$. Then, an *aperiodic timetable* for $\mathcal{N}$ is a vector $\pi \in \mathbb{Q}^{|V|}$ that satisfies

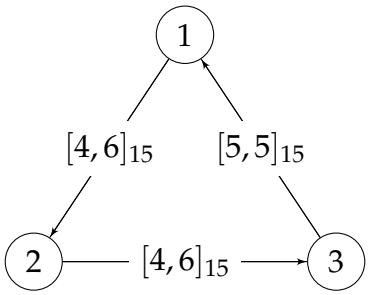
$$\pi_w - \pi_v \in [l_a, u_a] \; \forall \; a = (v, w) \in A$$

A vector $\pi \in \mathbb{Q}^{|V|}$ is called a *periodic timetable* for $\mathcal{N}$ with period T if there exists $p \in \mathbb{Z}^{|A|}$ such that for all $a = (v, w) \in A$ we have

$$l_a \leq \pi_w - \pi_v + p_a T \leq u_a$$

Periodicity is not only practical for passengers but also has operational purposes. For example, a whole tour of a train is supposed to be a multiple of the period. The following example illustrates this situation.

Example 2.3.3. An event network with cyclic constraints and possible solution $(\pi(1), \pi(2), \pi(3)) = (2, 7, 12)$.



The aperiodic timetable π or the periodic timetable $\pi \; mod \; T$ together with fixed positions of the events forms a slot. The acyclic timetabling problem is polynomially solvable using a shortest paths algorithm (Christian Liebchen, 2006), as is the slot construction. However, the *Periodic Event Scheduling Problem (PESP)*, i.e. the problem of finding a periodic timetable in the event network, is NP-complete for $T \geq 3$ (M. Odijk, 1994). The following MIP formulation additionally minimizes the weighted activity times.

$$\textbf{MIP-}PESP \text{ Minimize} \quad \sum_{a \in A} \omega_a (\pi_w - \pi_v + p_a T - l_a) \tag{2.2a}$$

$$\text{s.t.} \quad \pi_w - \pi_v + p_a T \leq u_a \quad \forall a = (v, w) \in A \tag{2.2b}$$

$$\pi_w - \pi_v + p_a T \geq l_a \quad \forall a = (v, w) \in A \tag{2.2c}$$

$$p_a \in \mathbb{Z} \quad \forall a \in A \tag{2.2d}$$

Solution approaches to tackle the PESP include satisfiability solving (Großmann et al., 2012b), the modulo simplex method by Nachtigall (1998), banch-and-cut of the MIP (Christian Liebchen, 2006) and mixtures of these (Borndörfer et al., 2020). When the

event network is a tree, then the PESP is polynomially solvable (Christian Liebchen, 2006). Lindner and Reisch (2020) extend this result by giving pseudo-polynomial-time dynamic programming algorithms if the event network has bounded tree- or branchwidth. Christian Liebchen (2008) put the PESP into practice and computed a cyclic timetable for the Berlin underground. In general, cyclic timetabling is most common for long-term planning of passenger trains.

2.4. Train Path Assignment Problem

In Section 2.2, we pointed out that a single slot can be constructed by applying a flow algorithm. If multiple slots are constructed simultaneously such that no two of them occupy the same infrastructure at the same time, this problem generalizes to the multi-commodity flow problem (A. Caprara et al., 2002). In general, this problem arises in long-term planning of both freight and passenger trains and we denote the problem of assigning a slot to each of maximally many trains $\mathcal{R}$ the *Train Path Assignment Problem (TPAP)*. Let q_r incidate whether or not a train r is assigned to a slot, $\mathbf{F}_r$ the incidence matrix of the train r and $\mathbf{C}$ the conflict matrix between arcs occupying the same infrastructure at the same time. Then, the TPAP can be modelled in the following MIP.

$$\begin{aligned}
\textbf{MIP-}\textit{TPAP}\ \text{Maximize} \quad & \sum_{r\in\mathcal{R}} q_r && && && (2.3a)\\
\text{s.t.} \quad & \mathbf{F}_r y_r = q_r I_r && && \forall\, r \in \mathcal{R} && (2.3b)\\
& \mathbf{C}\mathbf{y} \leq \mathbf{1} && && && (2.3c)\\
& q_r \in \{0,1\} && && \forall\, r \in \mathcal{R} && (2.3d)\\
& y_r = (((x,t)_i)_{i=0}^{n})_r && && \forall\, r \in \mathcal{R} && (2.3e)\\
& (y_i)_r \in \{0,1\} && \forall\, (y_i)_r = ((x,t)_i)_r && \forall\, r \in \mathcal{R} && (2.3f)
\end{aligned}$$

A variant is that the number of trains is fixed and the traveling times of the slots are minimized (Alberto Caprara, 2015; Zhang et al., 2019). Even et al. (1975) proved that the multi-commodity flow problem is NP-complete even for two commodities. This complexity result holds for both variants. Therefore, studies that solve TPAP for traveling time minimization, schedule at most several hundred of trains, as Zhang et al. (2019) have pointed out. On the other hand, Nachtigall and Opitz, 2014 use column generation to solve the TPAP for maximizing the capacity utilization of freight trains in the east of Germany. Reisch, Großmann, Pöhle, et al. (2021) extend this work by a heuristic column general approach and solve the TPAP for capacity maximization for all freight trains in Germany, that is, more than 5000. Zhang et al. (2019) close the gap to periodic timetabling by incorporating constraints to generate a cyclic timetable in the TPAP model.

2.5. Timetable Robustness

So far, we have seen models and solution approaches to schedule as many trains as possible or to find schedules with minimal traveling times. A third objective in railway timetabling is the robustness against minor delays that occur stochastically in railway operations. That is, given a distribution of minor disturbances, the sum of expected delays is to be minimized. The means to achieve this goal are buffer times between train trips on the one hand and time supplements in the timetable of a train, on the other hand. When buffer times are sufficiently large, minor delays of a train trip will not be propagated to the consecutive trip whereas supplements enable a train to compensate for delays that have occured already.

Stochastically occuring delays can be modelled by adding the amount ζ of delay to the minimum traveling times from x to x' in the TPAP model or to the lower bounds on the activities l in the PESP model. The vector ζ is referred to as a *scenario*. The scenario $\hat{\zeta} \in Z$ without any delays is called the *nominal scenario*. In (strict) robust optimization, the generated timetable needs to be feasible for every possible scenario ζ in a set Z of plausible delay scenarios (Goerigk and Schöbel, 2010). Let F be the constraints, x be the variable and f the objective. Then, in its most general form, a (strict) robust optimization problem reads as follows.

$$\textit{strict}\textbf{-robustness Minimize } f(x) \tag{2.4a}$$
$$\text{s.t. } F(x,\zeta) \leq 0 \qquad \forall \zeta \in Z \tag{2.4b}$$

Since this modelling is very conservative, Fischetti and Monaci (2009) introduce the concept of light robustness for train timetabling where exceedances γ_i of a constraint i are minimized in the objective function. Let z^* be the optimal value of the nominal problem and δ a parameter restricting the deviation from the optimal solution value of the nominal scenario. Then, the light robustness problem reads as follows.

$$\textit{light}\textbf{-robustness Minimize } \sum \gamma_i \tag{2.5a}$$
$$\text{s.t. } F(x,\hat{\zeta}) \leq 0 \tag{2.5b}$$
$$f(x,\hat{\zeta}) \leq (1+\delta)z^* \tag{2.5c}$$
$$F_i(x,\zeta) \leq \gamma_i \qquad \forall i\, \forall \zeta \in Z \tag{2.5d}$$
$$\gamma \geq 0 \tag{2.5e}$$

Schöbel and Kratz (2009) apply the robust optimization to the aperiodic timetabling in an bi-criteria approach compromising robustness and travelling times. Furthermore, there are studies where the robust timetable is not generated but merely modified. For instance, Maróti (2017) applies stochastic programming to find an optimal allocation of buffer and supplement times of a given reference timetable and applies it to a 1-hour timetable of the whole Netherlands Railways (NS).

Finally, there is a number of approaches that consist of evaluations of the robustness of a modified timetable only. Such approaches include both simulation (Curchod, 2007), and analytical computations (Huisman and Boucherie, 2001) to derive the expected amount of propagated delays in a timetable with respect to a distribution of occurring delays. Reisch and Kliewer (2020) close the gap to robust timetable modification by introducing black-box optimization rules for these evaluation approaches with the aim of adjusting the timetable such that the new one improves the objective of minimal delays.

2.6. Conclusion

In this state of the art overview, we considered different aspects of computer-aided railway timetabling. We presented the notion of a slot which is a timetable for a single train and that it can be computed in polynomial time. Furthermore, we stated the difference between periodic and aperiodic timetables and showed that the periodic problem modelled as the PESP, is NP-complete. Likewise, the complexity of scheduling multiple trains on a limited infrastructure, modelled as the TPAP, is NP-complete. Finally, we presented approaches that optimize railway timetables with respect to minimizing the sum of expected delays.

References

Borndörfer, Ralf, Niels Lindner, and Sarah Roth (2020). "A Concurrent Approach to the Periodic Event Scheduling Problem". In: *Journal of Rail Transport Planning & Management* 15, pp. 100–175. DOI: 10.1016/j.jrtpm.2019.100175.

Caprara, A. et al. (2002). "Solution of real-world train timetabling problems". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*.

Caprara, Alberto (2015). "Timetabling and assignment problems in railway planning and integer multicommodity flow". In: *Networks* 66.1, pp. 1–10.

Curchod, Anne (2007). *analyse de la stabilité d'horaires ferroviaires cadencés sur un réseau maillé: Bedienungshandbuch*. Lausanne: FASTA II.

Dahms, Florian et al. (2019). *Transforming automatic scheduling in a working application for a railway infrastructure manager*. Rail Norrköping Conference.

Even, S., A. Itai, and A. Shamir (1975). "On the complexity of time table and multicommodity flow problems". In: *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pp. 184–193.

Fischetti, Matteo and Michele Monaci (2009). "Light Robustness". In: *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Berlin, Heidelberg: Springer-Verlag, pp. 61–84. ISBN: 9783642054648. URL: https://doi.org/10.1007/978-3-642-05465-5_3.

Ford, L. R. and D. R. Fulkerson (1956). "Maximal Flow Through a Network". In: *Canadian Journal of Mathematics* 8, pp. 399–404. DOI: 10.4153/CJM-1956-045-5.

Goerigk, Marc and Anita Schöbel (2010). "An Empirical Analysis of Robustness Concepts for Timetabling". In: *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*. Ed. by Thomas Erlebach and Marco Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 100–113. ISBN: 978-3-939897-20-0. DOI: 10.4230/OASIcs.ATMOS.2010.100. URL: http://drops.dagstuhl.de/opus/volltexte/2010/2753.

Großmann, Peter et al. (2012b). "Solving Periodic Event Scheduling Problems with SAT". In: *Advanced Research in Applied Artificial Intelligence*. Ed. by He Jiang et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 166–175. ISBN: 978-3-642-31087-4.

Hansen, Ingo and Jörn Pachl (2014). *Railway Timetabling & Operations. Analysis - Modelling - Optimisation - Simulation - Performance Evaluation.*

Huisman, Tijs and Richard J. Boucherie (2001). "Running times on railway sections with heterogeneous train traffic". In: *Transportation Research Part B: Methodological* 35.3, pp. 271–292. ISSN: 0191-2615. DOI: https://doi.org/10.1016/S0191-2615(99)00051-X. URL: http://www.sciencedirect.com/science/article/pii/S019126159900051X.

Liebchen, Christian (2006). *Periodic Timetable Optimization in Public Transport*. dissertation.de.

— (2008). "The First Optimized Railway Timetable in Practice". In: *Transportation Science* 42.4, pp. 420–435. DOI: 10.1287/trsc.1080.0240. eprint: https://doi.org/10.1287/trsc.1080.0240. URL: https://doi.org/10.1287/trsc.1080.0240.

Lindner, Niels and Julian Reisch (2020). *Parameterized Complexity of Periodic Timetabling*. eng. Tech. rep. 20-15. Takustr. 7, 14195 Berlin: ZIB.

Maróti, Gábor (July 2017). "A branch-and-bound approach for robust railway timetabling". English. In: *Public Transport* 9.1-2, pp. 73–94. ISSN: 1866-749X. DOI: 10.1007/s12469-016-0143-x.

Nachtigall, Karl (1998). "Periodic Network Optimization and Fixed Interval Timetables". Habilitation thesis. Universität Hildesheim.

Nachtigall, Karl and Jens Opitz (2014). *Modelling and Solving a Train Path Assignment Model*. Proceedings of the International Conference on Operations Research, Aachen.

Odijk, Michiel (1994). *Construction of Periodic Timetables: A Cutting Plane Algorithm*. in: Technical Report, TU Delft.

Reisch, Julian, Peter Großmann, Daniel Pöhle, et al. (2021). "Conflict resolving – A local search algorithm for solving large scale conflict graphs in freight railway timetabling". In: *European Journal of Operational Research*. ISSN: 0377-2217. DOI: doi.org/10.1016/j.ejor.2021.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0377221721000084.

Reisch, Julian and Natalia Kliewer (2020). "Black-Box Optimization in Railway Simulations". In: *Operations Research Proceedings 2019*. Ed. by Janis S. Neufeld et al. Cham: Springer International Publishing, pp. 717–723. ISBN: 978-3-030-48439-2.

Schöbel, Anita and Albrecht Kratz (2009). "A Bicriteria Approach for Robust Timetabling". In: *Robust and Online Large-Scale Optimization: Models and Techniques for Trans-*

portation Systems. Ed. by Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 119–144. ISBN: 978-3-642-05465-5. DOI: 10.1007/978-3-642-05465-5_5. URL: https://doi.org/10.1007/978-3-642-05465-5_5.

Serafini, Paolo and Walter Ukovich (1989a). *A Mathematical Model for Periodic Scheduling Problems*. in: SIAM Journal on Discrete Mathematics, 2, pp. 550-581.

Zhang, Yongxiang et al. (2019). "Solving cyclic train timetabling problem through model reformulation: Extended time-space network construct and Alternating Direction Method of Multipliers methods". In: *Transportation Research Part B: Methodological* 128, pp. 344–379. ISSN: 0191-2615.

Part I.

Practical Applications of Automatic Railway Timetabling

3. Conflict Resolving - A Local Search Algorithm for Solving Large Scale Conflict Graphs in Freight Railway Timetabling

Chapter is omitted due to copyright.

4. Bestimmung der Kapazitätssteigerung durch Einführung der Mittelpufferkupplung und ep-Bremse

Chapter is omitted due to copyright.

Part II.

Algorithms and Computability in Timetable Generation

5. Conflict Resolving - A Maximum Independent Set Heuristics for Solving MaxSAT

Chapter is omitted due to copyright.

6. Stable Resolving - A Randomized Local Search Heuristic for MaxSAT

Abstract

Many problems from industrial applications and AI can be encoded as Maximum Satisfiability (MaxSAT). Often, it is more desirable to produce practicable results in very short time compared to optimal solutions after an arbitrary long computation time. In this paper, we propose Stable Resolving (SR), a novel randomized local search heuristic for MaxSAT with that aim. SR works for both weighted and unweighted instances. Starting from a feasible initial solution, the algorithm repeatedly performs the three steps of perturbation, improvements and solution checking. In the perturbation, the search space is explored at the cost of possibly worsening the current solution. The local improvements work by repeatedly flipping signs of variables in over-satisfied clauses. Finally, the algorithm performs a solution checking in a simulated annealing fashion. We compare our approach to state-of-the-art MaxSAT solvers and show by numerical experiments on benchmark instances from the annual MaxSAT competition that SR performs comparable on average and is even the best solver for particular problem instances.

Contents

6.1. Introduction

We consider the Constraint Satisfaction Problem of Maximum Satisfiability (MaxSAT). Many NP-hard optimization problems from applications in industry and AI can be encoded as MaxSAT and existing solution algorithms have proved to yield results that are competitive to domain specific solvers. The applications vary from periodic scheduling (Großmann et al., 2012c), to causal discovery (Hyttinen et al., 2014), Bayesian network structure learning (Berg, Järvisalo, and Malone, 2014), correlation clustering (Berg and Järvisalo, 2017), reasoning over bionetworks (Guerra and Lynce, 2012), probabilistic inference (Park, 2002) and many more. A MaxSAT encoding consists a Boolean formula that we assume to be in conjunctive normal form (CNF) which means that the literals are grouped in clauses where they are connected disjunctively (*or*) and the clauses are connected conjunctively (*and*).

Example. A Boolean formula in CNF: $F = (\neg x \vee \neg y) \wedge (\neg y \vee \neg z)$.

A literal is a variable together with a positive or negative sign. A clause is satisfied if at least one of its literals has the same sign as the variable in the solution. We also say the literal is *true* and denote the number of true literals in a clause its *stability*. An unsatisfied clause has a stability of zero. When a clause has a stability greater than 1, we say that the clause is over-satisfied. A solution is an assignment of the variables to *true* or $false$ and is called *feasible* if all hard clauses are satisfied. We assume the formula to consist of both hard clauses and (possibly weighted) soft clauses. The sum of (the weights of) satisfied soft clauses is the objective function value. Then, the task is to find a feasible solution maximizing the objective function value.

Example. Let $F = H_1 \wedge H_2 \wedge S_1 \wedge S_2 \wedge S_3$ where

$$
\begin{array}{ll}
H_1: \neg x \vee \neg y & \\
H_2: \neg y \vee \neg z & \\
S_1: x & weight(S_1) = 2 \\
S_2: y & weight(S_2) = 3 \\
S_3: z & weight(S_3) = 2
\end{array}
$$

are hard and soft clauses with according weights respectively. Then, the optimal solution of value 4 is $x = z = true$ and $y = false$.

Due to its generic form, almost any problem from combinatorial optimization and many optimization problems in AI can be encoded as MaxSAT and practice shows that this conversion often works well. In this paper, we propose a novel heuristic solution approach to the MaxSAT problem called Stable Resolving (SR). The aim is to solve even large problem instances with millions of clauses and variables within short time, that is, up to 60 seconds, to a practicable solution. To do so, SR repeatedly performs the three steps of perturbation, improvements and solution checking, starting from an initial feasible solution. In the perturbation, the search space is explored by satisfaction

of randomly picked unsatisfied (soft) clauses at the cost of other clauses becoming unsatisfied. More precisely, we consider the randomly picked clauses as hard clauses and call a SAT solver on them, together with the original hard clauses. If other, formerly satisfied soft clauses become unsatisfied by this perturbation, we write them in a list of unsatisfied candidate clauses. Then, in the improvement part, a local search technique is employed that builds on the clauses' stabilities. Starting with the first member of the list of unsatisfied candidate clauses, clauses with stability zero are being satisfied by flipping the sign of a randomly chosen variable. Flipping the sign of one of its variables increases the clause's stability by 1 but might cause other clauses to become unsatisfied. These unsatisfied clauses are added to the (local) search space and will be tried to be satisfied later. On the other hand, if flipping a variable's sign increases other clauses' stabilities to a number larger 1, that is, they become over-satisfied, they can have at least one literal falsified without becoming unsatisfied. This falsification can hence satisfy yet other clauses that contain the same variable with opposite sign and improve the objective function again. In this way, the local search space grows until all unsatisfied clauses have been tried to satisfy. Then, the improvement step ends and if the objective function value has decreased, the previous solution is restored. Else, newly unsatisfied clauses are added to the list of unsatisfied candidate clauses. As candidate clauses are only added when the objective function value increases, and one candidate is erased when it decreases, this list will eventually be empty. Then, the solution checking part begins. Here, a worsening of the objective value is allowed with a probability that decreases during the run of the algorithm.

The outline of the paper will be as follows. After a literature overview over existing approaches in Section 6.2, we explain the algorithm in detail in Section 6.3. In Section 6.4, we present and discuss our results on common benchmark instances and finally give a conclusion and outlook in Section 6.5.

6.2. Related Work

There are numerous solution approaches for the MaxSAT problem both exact and heuristic ones. Let us point out the differences between SR and other state-of-the-art MaxSAT solvers. In the 2019's MaxSAT competition[1], the solver Loandra performed best in the incomplete unweighted track. It combines a core-guided approach for finding a lower bound (Berg, Demirović, et al., 2019) and a linear algorithm for an upper bound. As the linear algorithm, the authors use LinSBPS (Demirović and Stuckey, 2019) that performs a neighborhood search in a complete algorithmic setting by repeatedly calling the SAT Solver glucose (Audemard and Simon, 2009). In contrast to LinSPBS, we only call glucose once at the beginning for an initial solution and for the perturbation of a solution but not in order to achieve an improvement. Moreover, we do not calculate lower bounds at all. The local search algorithms MaxRoster (a description can be found in Bacchus, Järvisalo, et al. (2019)) which is based on Ramp (Fan et al., 2016) and SATLike (Lei and Cai, 2018) which iteratively flips the sign of variables

[1] https://maxsat-evaluations.github.io/2019/index.html

that bring the best improvement work differently than our solver in the respect that they adapt weights of clauses in order to leave local optima. We, however, perturb a current solution for that purpose and instead of changing weights. (Max-)WalkSAT and GSAT (Selman et al., 1995) are local search approaches similar to SR in the sense that unsatisfied clauses are picked at random and one of their variables' sign is flipped. The difference to our approach is that SR searches a larger neighborhood with a more complex improvement heuristics based on stabilities. In fact, one can consider SR a large neighborhood search, as pursued in the OR world (cf. e.g. Pisinger and Ropke (2010)), with the difference that SR finds improvements in the neighborhood heuristically and without calling an exact solver whereas the repair procedure in large neighborhood searches often involve an exact solver. At the end of each iteration, SR checks the solution in a simulated annealing fashion. Simulated annealing with reset has been used also for MaxSAT (Hoos, 1996; Bouhmala, 2019). Finally, let us point out that the splitting of our algorithm into perturbation, improvement and solution checking was introduced for a state-of-the-art Maximum Independent Set (MIS) heuristic (Andrade et al., 2012) that in a previous work, we have been able to extend by a different improvement technique and simulated annealing solution checking in order to solve MaxSAT instances that have been transformed to MIS (Reisch, Großmann, and Kliewer, 2019). In contrast, in this paper we propose an algorithm that works directly on the Boolean formula.

6.3. Algorithm

The overall procedure of SR is shown in Algorithm 1. We first apply a SAT-based preprocessing on the formula. That is, we label the soft clauses meaning that each soft clause gets an additional variable l and will be considered a hard clause. In addition, for each label, we introduce a unit soft clause $\neg l$ with the weight the original soft clause had (Belov et al., 2013). For the obtained equivalent formula, we apply unit clause propagation and bounded variable elimination (cf. e.g. Davis and Putnam (1960)) on the hard clauses, as long as it is possible. Note that the label variables are excluded from the propagations since these operations are only sound for hard clauses. Then, for an initial feasible solution the SAT solver glucose (Audemard and Simon, 2009) is called.

The algorithm then repeatedly executes the three steps of perturbation, improvement and solution checking.

Algorithm 1 StableResolving()

```
Preprocess()
CalculateInitialSolution()
while timeout has not been reached do
    Perturb()
    StableImprove()
    CheckSolution()
end
```

Let us explain the single parts in greater detail. In the perturbation part shown in Algorithm 2, we explore the search space. More precisely, we first sample a random number k from the geometric distribution with parameter p and select k unsatisfied clauses uniformly at random. Then, we call the SAT solver glucose on all hard clauses and the selected clauses. Additionally, we give the previous solution as an initial solution to the solver in order to speed up the computation. If this formula is feasible, we have altered the solution, but maybe at the cost of a lower objective function value because formerly satisfied clauses are now unsatisfied. These unsatisfied clauses are added to the back of a list of *candidates* that potentially can be satisfied by improvements. We keep and update this list throughout the algorithm.

Algorithm 2 Perturb()

$k =$ random number where $\mathbb{P}[k = i] = p(1-p)^{i-1}$
$\mathcal{C} =$ set of k unsatisfied clauses picked uniformly at random
Call SAT solver on $\mathcal{C}$ and all hard clauses and overwrite the solution
Add newly unsatisfied clauses to the back of *candidates*

Example. Consider the example formula F from above. An initial feasible solution is given by all variables set to $false$. The perturbation might set $k = 1$, choose the unsatisfied clause $C = S_2$ and the SAT solver returns the feasible solution of $y = true$ and $x = z = false$. No clause gets unsatisfied by this step.

Remark. In some large instances from industrial applications, sampling a random unsatisfied clause is computationally expensive when all clauses are iterated through in order to detect the unsatisfied ones and sample among them. This is why we keep a superset of the unsatisfied clauses where every time a clause gets unsatisfied, it is added to. Moreover, we apply a heuristic in this superset and sample 1000 clause indices at random and only return if the corresponding clause indeed is unsatisfied. Only if all 1000 sampled clauses are satisfied, we iterate through the superset to find the unsatisfied clauses and sample among them.

Algorithm 3 StableImprove()

```
while candidates ≠ ∅ do
    C = pop first clause from candidates
    Init A = ∅ and 𝒞 = {C}
    while ∃v = variable picked uniformly at random in vars(𝒞) \ A do
        Flip sign of v and add v to A
        Add newly unsatisfied clauses to 𝒞
        Stab_{1→2} = set of clauses whose stability has grown to 2
        foreach S ∈ Stab_{1→2} do
            w = variable of second true literal in S
            if w is in no clause of stability 1 nor in A then
                Flip sign of w and add w to A
            end
        end
    end
    if objective function value has decreased then
        Revert flips of variables in A
    end
    else
        Add 𝒞 at the back of candidates
    end
end
```

In the improvement part shown in Algorithm 3, we iteratively pick a variable uniformly at random of an unsatisfied clause (at first from the candidates and later from the clauses that have been unsatisfied during this improvement step) and flip its sign. A flip might lead to other clauses becoming unsatisfied now and we store them in the set $\mathcal{C}$. Note that also hard clauses can become temporarily unsatisfied. On the other hand, there might be a set $Stab_{1\to 2}$ of clauses whose stability grows from 1 to 2 which means that there exists now a second true literal whose variable's sign can now be flipped without unsatisfying this clause. This optimization technique of considering variables in over-satisfied constraints is well-known in mathematical optimization (cf. e.g. Simplex Method (Dantzig, 1963)) and we apply it here as a local improvement heuristics. In our algorithm, the clauses in $Stab_{1\to 2}$ are iterated through and checked for such an improvement. When no more variables are found that can be flipped, either because $\mathcal{C}$ is empty or all variables from $\mathcal{C}$, denoted $vars(\mathcal{C})$, are flipped already, the improvement step ends. Either the objective function value has increased, then the now unsatisfied clauses are added to the candidates, or it has not and the flips, stored in A, are reverted. Note that the feasible solution remains feasible as the objective function value cannot increase when hard clauses have become unsatisfied. The improvement part ends when there are no more candidate clauses.

Remark. In some test instances, the set $\mathcal{C}$ monotonously grows and never shrinks because there are more new unsatisfied clauses than clauses that can either be sat-

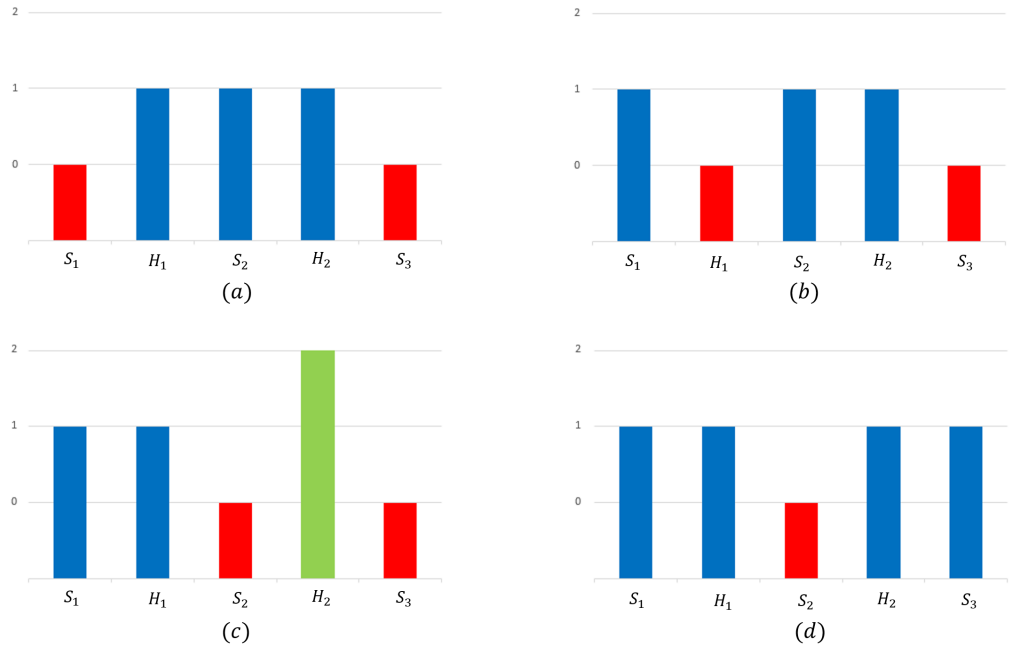


Figure 6.1.: Stabilities of clauses during an improvement step.

isfied or whose variables have all been considered for an improvement. In order to avoid that we spend too much time in a single local improvement step, we set an iterations limit of 25 for the inner while-loop.

Note that both while-loops terminate. For the outer one, candidate clauses are only added if the objective function value has increased which cannot be infinitely often as MaxSAT instances are always bounded. The inner one ends - besides the iterations limit - when A contains all variables.

Example. Consider the example formula F with solution

$$(x, y, z) = (false, true, false)$$

from above. The stabilities of the following steps are illustrated in Figure 6.1. S_1 is unsatisfied and might be the first candidate clause (a). Flipping the sign of its only variable x unsatisfies H_1 because y has been set to *true* in the perturbation already (b). Hence, H_1 gets added to $\mathcal{C}$ and x to A. Flipping the sign of one of the variables of H_1 (the only clause in $\mathcal{C}$) that has not already been flipped (i.e. that is not in A) means flipping y to $false$. Note that H_2 has now stability 2 and gets added to $Stab_{1\to 2}$ as both variables y and z are $false$ (c). The variable z is the second *true* literal that has been *true* before, so its sign gets flipped because no further clause is being unsatisfied by that flip. The improvement step ends with a objective function value that has increased from 3 to 4 (d).

Let us mention that during an improvement step (and after the perturbation), it is possible that formerly satisfied hard clauses become unsatisfied. Hard clauses have a weight greater than the sum of the weights of the soft clauses. Therefore, breaking hard clauses (without satisfying other formerly unsatisfied hard clauses) worsens the solution. In order to leave local optima, however, a worsening is possible in our algorithm - with decreasing probability according to the simulated annealing step, as will be explained in the remainder of this section.

Algorithm 4 CheckSolution()

```
if objective function value has increased to the best one ever seen then
    Save new best solution
end
else if objective function value has decreased then
    if number of iterations without improvement has exceeded m then
        Restore best solution
    end
    else
        Restore previous solution with probability exp(−prob)
    end
end
```

After the improvements we have arrived in a local optimum. The current solution might be of smaller objective function value than the previous solution from before this iteration of Algorithm 1 if the improvements could not compensate the perturbation. Still, we sometimes allow such a worsening in the simulated annealing approach shown in Algorithm 4 in order to be able to leave local optima. More precisely, we restore the previous solution if it had a better objective function value with a probability growing exponentially with a factor *prob* that decreases linearly during the course of the algorithm from 1 to 0 and represents the temperature of the simulated annealing. If, however, the number of iterations without an improvement exceeds a parameter m, we reset to the best solution ever seen. When SR terminates, this best solution is returned.

6.4. Experimental Results

We have applied SR to problem instances and compared it to results that are taken from the 2019's MaxSAT competition[2]. The instances encode various industrial applications' and theoretical problems, such as scheduling, fault diagnosis, tree-width computation, max clique problems, causal discovery, Ramsey number approximation and many more. An overview of the competing solvers can be found in Bacchus, Järvisalo, et al. (2019). For all calculations, we set the parameters for the geometric distribution and maximum steps in SR to $p = 0.75$ and $m = 1000$, respectively, because they yield

[2]We have submitted SR to the 2020's MaxSAT competition.

Table 6.1.: Sum of scores by solver on unweighted instances

Loandra	LinSBPS 2018	SR	SATLike*	Open WBO g	sls mcs*	sls mcs lsu*	Open WBO ms
251.7327	238.3298	231.1436	227.4589	204.1828	202.7803	202.7158	190.9274

Table 6.2.: Sum of scores by solver on weighted instances

Loandra	236.2272
TT Open WBO Inc*	233.4784
LinSBPS2018	231.6581
Open WBO Inc (inc bmo satlike)*	220.3607
Open WBO Inc (inc bmo complete)*	218.6454
SR	213.3262
Open WBO g*	212.1081
SATLike*	210.6802
sls mcs2*	203.1498
Open WBO ms*	194.5451
sls mcs*	191.4503
uwrmaxsat inc*	190.7841

the best results on average. We performed all computations on an Intel Core i7-8700K and with a time limit of 60 seconds. Note that if a solver from the MaxSAT competition yields worse results on our machine than in the results of the 2019's MaxSAT competition where computations were performed on the StarExec Cluster[3], we include the better results for the analysis here. We mark such solvers with an asterisk*.

Table 6.1 and Table 6.2 show the sum of scores of the competing solvers on the unweighted and weighted benchmark instances, respectively, from the incomplete track of the MaxSAT competition against the scores of SR. The score of a solver on an instance is calculated in the following way. Maximizing the sum (of weights) of satisfied soft clauses is equivalent to minimizing the sum (of weights) of unsatisfied soft clauses, which is denoted by the *gap*. The score of a solver on an instance is the fraction of the best gap known divided by the gap of the particular solver. If a solver's solution violates a hard clause, its score is zero and its gap infinity.

Example. Consider the example above. In the optimal solution, only S_2 is unsatisfied which yields the optimal gap of 3. If solver A has achieved this optimum, solvers B and C satisfy only S_2 and not S_1 and S_3, then their scores are $3/4$ while solver A has the maximal score of 1 on this instance.

[3]https://www.starexec.org/starexec/public/about.jsp

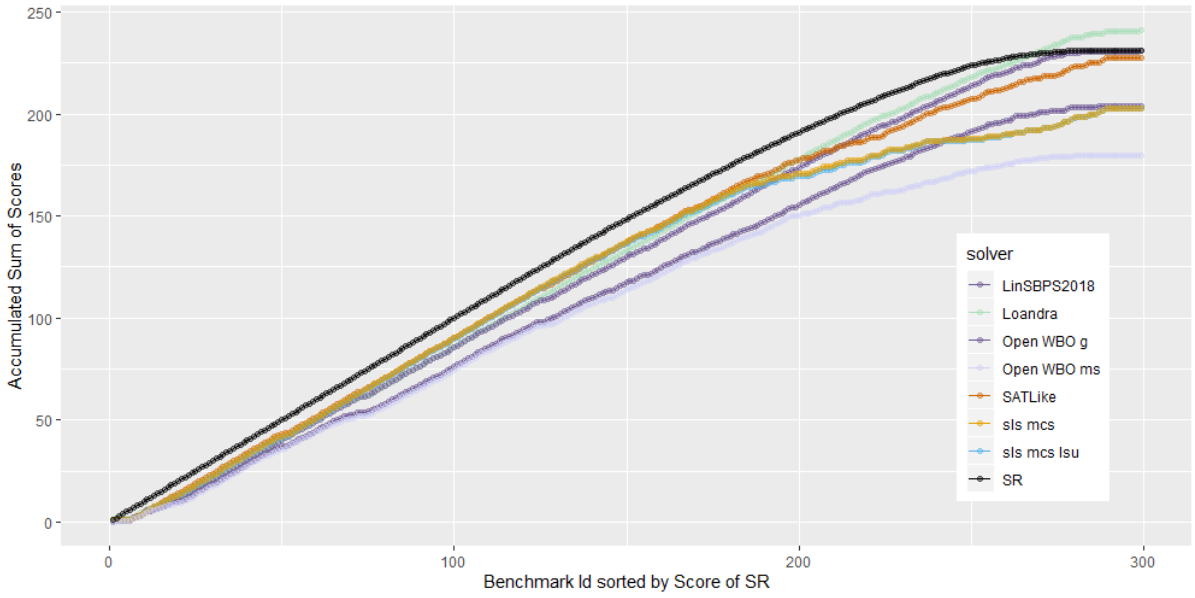


Figure 6.2.: Accumulated sum of scores of unweighted instances after 60 sec computation time

The score hence reflects the ratio of the achieved result to the optimal (or best known) one. In Figure 6.2 and Figure 6.3, we see the accumulated sum of scores of the single instances, ordered by SR's scores and grouped by the competing solvers for the unweighted and weighted instances, respectively. The figures illustrate that SR (black) has the highest sum of scores on a large subset of instances. Counting all instances, including those where SR has low scores, we conclude that SR still has a competitive performance. More precisely, in 210 and 179 of the 299 unweighted and 297 weighted instances, SR has a score at least the mean of the other solvers. Furthermore, SR performs especially well on the unweighted instances, in comparison to the other solvers.

What is more, SR often has the best result among all solvers for particular instances. We observe that in the unweighted case, SR performs especially well on instances from *atcos*, *extension enforcement* and *set covering*. In the weighted case, SR is best on many instances encoding the *Minimum Weight Dominating Set Problem*. See Tables 6.3 and 6.4 for complete lists of such instances in the unweighted and weighted case, respectively. For a better comparison, we include a column showing the gaps of the winning solver Loandra, as well.

6.5. Conclusion and Outlook

In this paper, we have proposed a novel local search algorithm for solving large MaxSAT problems in short time. We could prove by numeric experiments on benchmark instances encoding problems from combinatorial optimization and AI that our algorithm yields results that are comparable to and for some problem families even better

Table 6.3.: Gaps of unweighted instances where SR performs best

	Benchmark	SR	Loandra
1	aes/sbox-8.wcnf.gz	443	690
2	atcoss/mesat/atcoss-mesat-04.wcnf.gz	97	Inf
3	atcoss/mesat/atcoss-mesat-05.wcnf.gz	74	Inf
4	atcoss/mesat/atcoss-mesat-10.wcnf.gz	32	40
5	atcoss/mesat/atcoss-mesat-18.wcnf.gz	80	Inf
6	atcoss/sugar/atcoss-sugar-15.wcnf.gz	133	Inf
7	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-1-10-0.wcnf.gz	7	18
8	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-2-10-2.wcnf.gz	12	20
9	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-3-10-1.wcnf.gz	8	9
10	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-3-10-4.wcnf.gz	10	16
11	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-4-10-1.wcnf.gz	7	17
12	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-4-10-2.wcnf.gz	6	13
13	extension-enforcement/extension-enforcement-non-strict-stb-200-0.05-4-10-4.wcnf.gz	8	18
14	min-fill/MinFill-R3-miles1000.wcnf.gz	3634	3755
15	optic/gen-cvc-add7to3-9999.wcnf.gz	197	204
16	pseudoBoolean/garden/normalized-g100x100.opb.msat.wcnf.gz	2163	2526
17	railway-transport/d4.wcnf.gz	8296	8524
18	SeanSafarpour/wb-4m8s1.dimacs.filtered.wcnf.gz	58	282
19	SeanSafarpour/wb-4m8s4.dimacs.filtered.wcnf.gz	220	230
20	set-covering/crafted/scpclr/scpclr13-maxsat.wcnf.gz	27	28
21	set-covering/crafted/scpcyc/scpcyc07-maxsat.wcnf.gz	145	149
22	set-covering/crafted/scpcyc/scpcyc08-maxsat.wcnf.gz	363	390
23	set-covering/crafted/scpcyc/scpcyc09-maxsat.wcnf.gz	835	972
24	set-covering/crafted/scpcyc/scpcyc10-maxsat.wcnf.gz	1967	2242
25	set-covering/crafted/scpcyc/scpcyc11-maxsat.wcnf.gz	4771	5623
26	uaq/uaq-ppr-nr200-nc66-n5-k2-rpp4-ppr12-plb100.wcnf.gz	75	78
27	xai-mindset2/liver-disorder.wcnf.gz	316	318

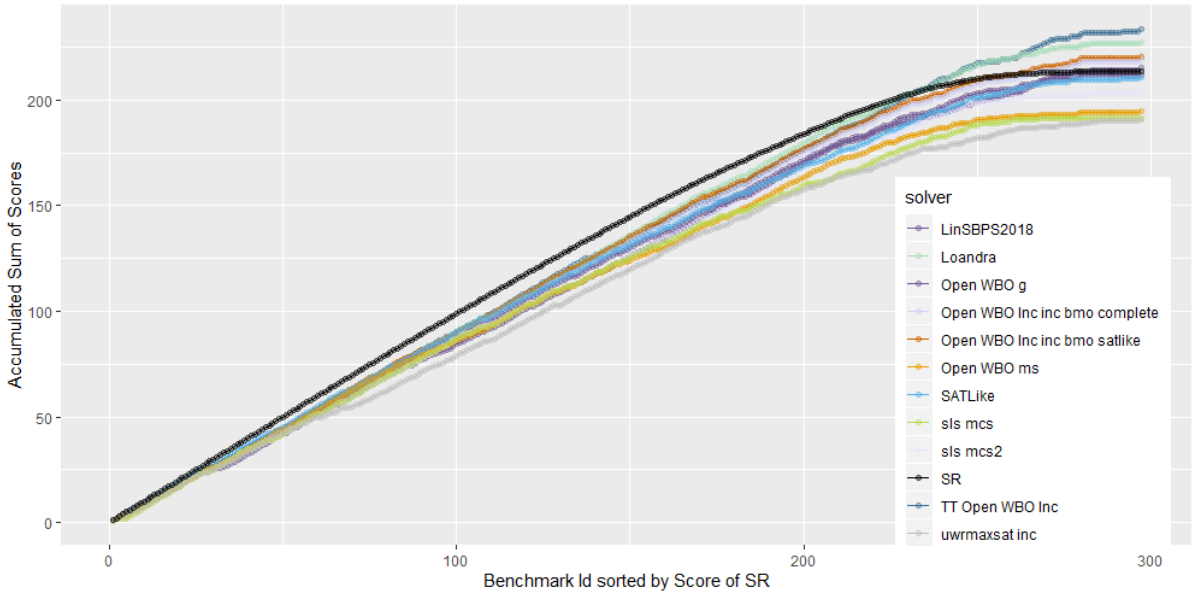


Figure 6.3.: Accumulated sum of scores of weighted instances after 60 sec computation time

Table 6.4.: Gaps of weighted instances where SR performs best

	Benchmark	SR	Loandra
1	causal-discovery/causal-Water-10-1000.wcnf.gz	11339025	16041455
2	causal-discovery/causal-Wdbc-8-569.wcnf.gz	1446339	2541316
3	correlation-clustering/Rounded-CorrelationClustering-Vowel-BINARY-N740-D0.200.wcnf.gz	120199215	130874895
4	correlation-clustering/Rounded-CorrelationClustering-Vowel-BINARY-N760-D0.200.wcnf.gz	120800405	132256968
5	drmx-cryptogen/geffe128-7.wcnf.gz	812	846
6	min-width/MinWidthCB-mitdbsample-100-43-1k-5s-2t-5.wcnf.gz	32010	32200
7	min-width/MinWidthCB-mitdbsample-200-64-1k-2s-1t-4.wcnf.gz	76975	78325
8	min-width/MinWidthCB-mitdbsample-300-43-1k-6s-1t-8.wcnf.gz	45780	45825
9	MinimumWeightDominatingSetProblem/delaunay-n24.wcnf.gz	304532225	350820532
10	MinimumWeightDominatingSetProblem/hugebubbles-00020.wcnf.gz	694937186	753286458
11	MinimumWeightDominatingSetProblem/inf-road-usa.wcnf.gz	840126999	903206743
12	MinimumWeightDominatingSetProblem/sc-rel9.wcnf.gz	15590036	16746750
13	MinimumWeightDominatingSetProblem/web-wikipedia2009.wcnf.gz	28120892	37674803
14	pseudoBoolean/miplib/normalized-mps-v2-20-10-p0548.opb.msat.wcnf.gz	12451	25494
15	spot5/log/1401.wcsp.log.wcnf.gz	463106	469110
16	spot5/log/1407.wcsp.log.wcnf.gz	459591	465638

than state-of-the-art solvers.

As a possible prospect, we aim at developing more sophisticated improvement methods that take into account not single over-satisfied clauses but sets of such. Also, we can think of caching unsuccessful local improvements so that they will never be performed a second time. Finally, we want to analyse the different components of our algorithm by replacing each of the perturbation, stable improvements and simulated annealing by a naive technique. This will give an insight into the contribution of each component to the solvers performance.

References

Andrade, Diogo Vieira, Mauricio G. C. Resende, and Renato Fonseca F. Werneck (2012). "Fast local search for the maximum independent set problem". In: *J. Heuristics* 18.4, pp. 525–547.

Audemard, Gilles and Laurent Simon (2009). "Predicting Learnt Clauses Quality in Modern SAT Solvers". In: *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. IJCAI'09. Pasadena, California, USA, pp. 399–404.

Bacchus, Fahiem, Matti Järvisalo, and Ruben Martins (Sept. 2019). "MaxSAT Evaluation 2018: New Developments and Detailed Results". In: *Journal on Satisfiability, Boolean Modeling and Computation* 11, pp. 99–131. DOI: 10.3233/SAT190119.

Belov, Anton, António Morgado, and Joao Marques-Silva (2013). "SAT-Based Preprocessing for MaxSAT". In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by Ken McMillan, Aart Middeldorp, and Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 96–111. ISBN: 978-3-642-45221-5.

Berg, Jeremias, Emir Demirović, and Peter J. Stuckey (2019). "Core-Boosted Linear Search for Incomplete MaxSAT". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Ed. by Louis-Martin Rousseau and Kostas Stergiou. Cham: Springer International Publishing, pp. 39–56. ISBN: 978-3-030-19212-9.

Berg, Jeremias and Matti Järvisalo (2017). "Cost-optimal constrained correlation clustering via weighted partial Maximum Satisfiability". In: *Artificial Intelligence* 244. Combining Constraint Solving with Mining and Learning, pp. 110–142. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2015.07.001.

Berg, Jeremias, Matti Järvisalo, and Brandon Malone (22–25 Apr 2014). "Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, pp. 86–95.

Bouhmala, Noureddine (Feb. 2019). "Combining simulated annealing with local search heuristic for Max-SAT". In: *Journal of Heuristics* 25.1, pp. 47–69.

Dantzig, George B. (1963). "Linear Programming and Extensions". In: Princeton University Press, Princeton.

Davis, Martin and Hilary Putnam (July 1960). "A Computing Procedure for Quantification Theory". In: *J. ACM* 7.3, pp. 201–215. ISSN: 0004-5411. DOI: 10.1145/321033.321034.

Demirović, Emir and Peter J. Stuckey (2019). "Techniques Inspired by Local Search for Incomplete MaxSAT and the Linear Algorithm: Varying Resolution and Solution-Guided Search". In: *Principles and Practice of Constraint Programming*. Ed. by Thomas Schiex and Simon de Givry. Cham: Springer International Publishing, pp. 177–194. ISBN: 978-3-030-30048-7.

Fan, Yi et al. (2016). "Ramp: A Local Search Solver based on Make-positive Variables". In: *MaxSAT Evaluation*.

Großmann, Peter et al. (2012c). *Solving Periodic Event Scheduling Problems with SAT*. in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 166-175, Springer.

Guerra, João and Inês Lynce (2012). "Reasoning over Biological Networks Using Maximum Satisfiability". In: *Principles and Practice of Constraint Programming*. Ed. by Michela Milano. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 941–956. ISBN: 978-3-642-33558-7.

Hoos, Holger H. (1996). "Solving hard combinatorial problems with GSAT — A case study". In: *KI-96: Advances in Artificial Intelligence*. Ed. by Günther Görz and Steffen Hölldobler. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 107–119. ISBN: 978-3-540-70669-4.

Hyttinen, Antti, Frederick Eberhardt, and Matti Järvisalo (2014). "Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming". In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pp. 340–349.

Lei, Zhendong and Shaowei Cai (July 2018). "Solving (Weighted) Partial MaxSAT by Dynamic Local Search for SAT". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 1346–1352. DOI: 10.24963/ijcai.2018/187.

Park, James D. (2002). "Using Weighted Max-SAT Engines to Solve MPE". In: *Proc. 18th Nat'l Conf. Artificial Intelligence*, pp. 682–687.

Pisinger, David and Stefan Ropke (2010). "Large Neighborhood Search". In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. Boston, MA: Springer US, pp. 399–419.

Reisch, Julian, Peter Großmann, and Natalia Kliewer (2019). "Conflict Resolving - A Maximum Independent Set Heuristics for Solving MaxSAT". In: *Proceedings of the 22nd International Multiconference Information Society* 1, pp. 67–71.

Selman, Bart, Henry Kautz, and Bram Cohen (1995). "Local Search Strategies for Satisfiability Testing". In: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 521–532.

7. Parameterized Complexity of Periodic Timetabling

Abstract

Public transportation networks are typically operated with a periodic timetable. The Periodic Event Scheduling Problem (PESP) is the standard mathematical modelling tool for periodic timetabling. Since PESP can be solved in linear time on trees, it is a natural question to ask whether there are polynomial-time algorithms for input networks of bounded treewidth. We show that deciding the feasibility of a PESP instance is NP-hard even when the treewidth is 2, the branchwidth is 2, or the carvingwidth is 3. Analogous results hold for the optimization of reduced PESP instances, where the feasibility problem is trivial. To complete the picture, we present two pseudo-polynomial-time dynamic programming algorithms solving PESP on input networks with bounded tree- or branchwidth. We further analyze the parameterized complexity of PESP with bounded cyclomatic number, diameter, or vertex cover number. For event-activity networks with a special – but standard – structure, we give explicit and sharp bounds on the branchwidth in terms of the maximum degree and the carvingwidth of an underlying line network. Finally, we investigate several parameters on the smallest instance of the benchmarking library PESPlib.

Contents

7.1. Introduction

Creating and optimizing timetables is substantial for planning and operating public transportation networks. As well as in local traffic as in long-distance train networks, timetables are often periodic, i.e., the schedule of trips repeats after a certain period time T, e.g., 60 minutes.

Mathematically, periodic timetabling is captured by the *Periodic Event Scheduling Problem* (PESP, Serafini and Ukovich, 1989b). The idea behind PESP is to model arrival and departure events of trips in a public transportation network as vertices (*events*) of a directed graph. Dependencies between events, such as driving of a vehicle or changing at a station, are modeled as arcs (*activities*) connecting pairs of events. These activities come with restrictions on their duration, e.g., driving from one station to the next might take at least 7 minutes. Then, a solution to PESP is an assignment of times in $[0, T)$ to each event (a *periodic timetable*) such that the activity duration restrictions are respected. We refer to Section 7.2 for rigorous formulations.

Deciding whether a periodic timetable exists is an NP-complete problem, even if $T \geq 3$ is not considered as part of the input. This result can be proved by a polynomial-time reduction of T-VERTEX COLORING, where a T-coloring of a graph corresponds to event times in $\{0, 1, \ldots, T-1\}$ (M. A. Odijk, 1994). This suggests a close relationship between PESP and coloring problems. In fact, both PESP and VERTEX COLORING are solvable in linear time on trees, regardless of T. Furthermore, for any T, deciding if a graph admits a T-coloring is fixed-parameter tractable when parameterized by treewidth. More precisely, there is a function f such that for a given graph G on n vertices with a nice tree decomposition of treewidth $\leq k$ and a natural number T, there is an $O(f(k) \cdot n)$ algorithm deciding the T-VERTEX COLORING problem on G (Arnborg and Proskurowski, 1989).

We show in Section 7.3 that no such result holds for PESP: Even if the event-activity network has treewidth 2 or branchwidth 2, i.e., every connected component is series-parallel, it is an NP-complete problem to decide whether a feasible periodic timetable exists. When considering *reduced* PESP instances, where the activities carry only lower bounds, but no (non-trivial) upper bounds, we prove that it is NP-complete to decide if there exists a periodic timetable whose weighted periodic slack is below a given threshold. Both proofs work by a reduction of the SUBSET SUM problem. As a byproduct, we also obtain an NP-hardness result on networks of carvingwidth 3. As a consequence, if P $\neq$ NP, then there are only pseudo-polynomial time algorithms available for both the feasibility and the reduced optimality problem. We give two such algorithms based on dynamic programming in Section 7.4, one in terms of a branch decomposition, and the other one in terms of a nice tree decomposition.

In Section 7.5, we prove that the feasibility version of PESP is fixed-parameter tractable when parameterized by the cyclomatic number, i.e., the dimension of the cycle space of the event-activity network. For computing periodic timetables with minimum weighted periodic slack, we give a polynomial-time algorithm when the cyclomatic number is bounded. We further discuss the diameter as a parameter. Some generalizations of the VERTEX COLORING problem like, e.g., $L(2,1)$-LABELING, are also NP-complete on graphs of treewidth ≥ 2, but are fixed-parameter tractable when

parameterized by the vertex cover number, i.e., the cardinality of a minimum vertex cover (Fiala et al., 2011). We prove that deciding the feasibility of a PESP instance is $W[1]$-hard when parameterized by the vertex cover number.

As PESP instances arising from public transportation networks typically have a special structure, we show in Section 7.6 that on this type of event-activity networks, the branchwidth can be related to invariants of an underlying line network: Roughly speaking, the number of lines at a station of a public transport network is a (sharp) lower bound on the branchwidth of the event-activity network. The carvingwidth of the often planar line network provides an upper bound, which is also sharp. Finally, we consider the smallest instance R1L1 of the PESP benchmarking library PESPlib, and use the relation to line networks in order to compute bounds on the width parameters discussed in this paper.

The presentation finishes with a few concluding remarks in Section 7.7.

7.2. The Periodic Event Scheduling Problem

The *Periodic Event Scheduling Problem* (PESP) was introduced in (Serafini and Ukovich, 1989b). PESP instances comprise the following ingredients:

- a directed graph G (often called *event-activity network*) with vertex set $V(G)$ (*events*) and arc set $A(G)$ (*activities*),
- a *period time* $T \in \mathbb{N}$,
- *lower bounds* $\ell \in \mathbb{Z}_{\geq 0}^{A(G)}$, $\ell < T$,
- *upper bounds* $u \in \mathbb{Z}_{\geq 0}^{A(G)}$, $u \geq \ell$,
- *weights* $w \in \mathbb{Q}_{\geq 0}^{A(G)}$.

In the application of periodic timetabling in public transport, the events are typically arrivals or departures of a line at a station, and activities model driving between stations, dwelling at a station, passenger transfers, or safety constraint such as minimum distances between vehicles (Christian Liebchen and Möhring, 2007). The weights often reflect the number of passengers using an activity.

Definition 7.2.1. Given (G, T, ℓ, u) as above, a *periodic timetable* is a vector $\pi \in [0, T)^{V(G)}$ such that there exists a *periodic tension* $x \in \mathbb{R}_{\geq 0}^{A(G)}$ satisfying

$$\forall ij \in A(G): \quad \ell_{ij} \leq x_{ij} \leq u_{ij} \quad \text{and} \quad \pi_j - \pi_i \equiv x_{ij} \mod T. \tag{7.1}$$

Intuitively, π gives the cyclic order of the events, and x corresponds to the duration of the activities. If there exists a periodic timetable π, then a periodic tension can be computed by

$$x_{ij} := [\pi_j - \pi_i - \ell_{ij}]_T + \ell_{ij}, \quad ij \in A(G), \tag{7.2}$$

where $[\cdot]_T$ denotes the modulo-T-operator taking values in $[0, T)$. Conversely, from a vector $x \in \mathbb{R}_{\geq 0}^{A(G)}$ with $\ell \leq x \leq u$, one can construct a periodic timetable with tension x by a graph traversal, see also Lemma 7.2.7.

In terms of the incidence matrix $B \in \{-1, 0, 1\}^{V(G) \times A(G)}$, condition (7.1) can be rewritten as

$$\ell \leq x \leq u \quad \text{and} \quad B^t \pi \equiv x \mod T.$$

Since B and hence are B^t are totally unimodular (Schrijver, 1986, Example 19.2) and the bounds ℓ, u are integer, it follows that if a periodic timetable exists, then there is also an integer timetable with an integer periodic tension. However, in general, it is not at all clear that a periodic timetable exists:

Definition 7.2.2 (T-PESP-FEASIBILITY)**.** Given a tuple (G, T, ℓ, u) as above, decide if there exists a periodic timetable π.

Theorem 7.2.3 (M. A. Odijk, 1994)**.** T-PESP-FEASIBILITY *is* NP*-complete for fixed* $T \geq 3$.

Proof. It is clear that T-PESP-FEASIBILITY is in NP, as a feasible timetable π with periodic tension x serves as certificate. We recall the proof in order to emphasize the natural relationship between PESP and VERTEX COLORING. Fix $T \geq 3$. Given an undirected graph H, construct a T-PESP-FEASIBILITY instance (G, T, ℓ, u) as follows: G is obtained from H by arbitrarily directing the edges, and set $\ell := 1$, $u := T - 1$. Then, H admits a T-coloring if and only if (G, T, ℓ, u) has a periodic timetable π. Namely, if $f : V(H) \to \{0, \dots, T-1\}$ is a T-coloring, then setting $\pi_i := f(i)$ for all $i \in V(G)$ is a periodic timetable, as for every arc $ij \in A(G)$ then holds $x_{ij} = [\pi_j - \pi_i]_T = [f(j) - f(i)]_T \in [1, T-1]$. Vice versa, if the PESP instance is feasible, then there is an integer timetable, giving rise to a T-coloring. □

So far, we have neglected the weight vector $w \in \mathbb{Q}_{\geq 0}^{A(G)}$. The weights come into play in the optimization variant of PESP, which we state as its corresponding decision version. If x is a periodic tension, we call $y := x - \ell \geq 0$ the *periodic slack*.

Definition 7.2.4 (T-PESP-OPTIMALITY)**.** Given (G, T, ℓ, u, w) as above and a number M, find a periodic timetable π with periodic slack y such that $w^t y \leq M$.

Minimizing the weighted periodic slack, or equivalently, the weighted periodic tension, can be interpreted as minimizing the total travel time of all passengers in a public transportation network. Clearly, T-PESP-OPTIMALITY is an NP-hard optimization problem. However, in many non-railway public transport networks, minimum distances are neglected for planning, and the driving and dwelling times of vehicles have a rather small span, so that they can be assumed as fixed. Contracting the corresponding activities yields a graph where only transfer activities remain, and these have typically no restrictions on their durations in the sense that lower and upper bounds differ by at least $T - 1$. This motivates the following specialization of PESP, sometimes called *reduced PESP*:

Definition 7.2.5 (T-RPESP-OPTIMALITY)**.** Given (G, T, ℓ, u, w) as above with $u \geq \ell + T - 1$ and a number M, find a periodic timetable π with periodic slack y such that $w^t y \leq M$.

Note that the feasibility problem is trivial to solve: *Any* integral vector $\pi \in [0,T)^{V(G)}$ is a periodic timetable, because for any acitivity $ij \in A(G)$,

$$\ell_{ij} \le x_{ij} = [\pi_j - \pi_i - \ell_{ij}]_T + \ell_{ij} \le T - 1 + \ell_{ij} \le u_{ij}.$$

Theorem 7.2.6 (Nachtigall, 1993)**.** *T-RPESP-Optimality is NP-hard for any fixed $T \ge 3$.*

Proof. We adapt the proof of Nachtigall to our notions and notations. Fix some period time $T \ge 3$. We reduce T-PESP-Feasibility to T-RPESP-Optimality. Let (G,T,ℓ,u) be a T-PESP-Feasibility instance. Without loss of generality, assume that $u - \ell < T$, because if the instance is feasible, then there is a periodic tension x satisfying $x < \ell + T$ by (7.2). Add to each arc $a \in A(G)$ from i to j a reverse copy $\bar{a}$ with $\ell_{\bar{a}} := [-u_a]_T$. Set all weights w to 1. For this T-RPESP-Optimality instance, let π be a periodic timetable with tension x as defined in (7.2). Then for any original arc a holds $\ell_a \le x_a < \ell_a + T$ and $x_a \equiv -x_{\bar{a}} \mod T$. For the slacks y_a and $y_{\bar{a}}$, we obtain

$$y_a + y_{\bar{a}} = [x_a - \ell_a]_T + [u_a - x_a]_T.$$

Since $0 \le x_a - \ell_a \le u_a - \ell_a < T$ and $-T < \ell_a - x_a \le u_a - x_a \le u_a - \ell_a < T$,

$$y_a + y_{\bar{a}} = \begin{cases} u_a - \ell_a & \text{if } u_a - x_a \ge 0, \\ u_a - \ell_a + T & \text{if } u_a - x_a < 0. \end{cases}$$

In particular, for the weighted slack of the T-RPESP-Optimality instance holds

$$\sum_{a \in A(G)} (y_a + y_{\bar{a}}) \ge \sum_{a \in A(G)} (u_a - \ell_a),$$

and equality holds if and only if $x_a \le u_a$ for all $a \in A(G)$. This means that the T-PESP-Feasibility instance is feasible if and only if the described T-RPESP-Optimality instance has weighted periodic slack at most $M := \sum_{a \in A(G)} (u_a - \ell_a)$. □

We turn now to simple algorithms for T-PESP-Optimality. Consider at first instances where undirecting the event-activity network results in a tree (shortly, G is a tree):

Lemma 7.2.7. *If G is a tree on n vertices, then T-PESP-Optimality on (G,T,ℓ,u,w) can be solved in $O(n)$ time. Morever, ℓ is an optimal periodic tension, and the minimum weighted periodic slack is 0.*

Proof. If undirecting G results in a tree on n vertices, then the transpose B^t of the incidence matrix B of G is a $(n-1) \times n$ matrix of full rank $n-1$. In particular, $B^t\pi = \ell$ has a solution over $\mathbb{Z}$, and reducing modulo T gives a feasible periodic timetable π^* with periodic tension ℓ and hence weighted periodic slack 0.

Avoiding linear algebra, π^* can as well be obtained by traversing the tree, starting with $\pi^*_v = 0$ at an initial vertex v and setting $\pi^*_j := [\pi^*_i + \ell_{ij}]_T$ when traversing $ij \in A(G)$, and $\pi^*_j := [\pi^*_i - \ell_{ij}]_T$ if $ji \in A(G)$ is traversed. A depth-first traversal takes $O(|V(G)| + |A(G)|) = O(n)$ time. □

On general networks, there are several ways to give naive exponential-time algorithms for T-PESP-OPTIMALITY:

Lemma 7.2.8. *On instances (G, T, ℓ, u, w) with n events and m activities, T-PESP-OPTIMALITY can be solved in*

1. $O^*(T^{n-1})$, *or*
2. $O^*(2^{n-1}n^{n-2})$, *or*
3. $O^*(3^m)$ *time,*

where $O^*(\cdot)$ means $O(\cdot)$ ignoring polynomial factors.

Proof.

1. Enumerate all T^n integral vectors in $[0,T)^{V(G)}$, compute x by (7.2), and check the bounds. This is an $O(mT^n)$ algorithm. If π is a periodic timetable, then for any $d \in \mathbb{R}$, π' defined by $\pi'_i := [\pi_i + d]_T$ for all $i \in V(G)$ is a periodic timetable with the same periodic tension. In particular, only T^{n-1} vectors have to be enumerated.

2. By Nachtigall (1998), if G is weakly connected and the instance is feasible, there is an optimal periodic tension x^* and a spanning tree F of G such that $x^*_a \in \{\ell_a, u_a\}$ for all $a \in A(F)$. Enumerate all $O(n^{n-2})$ spanning trees F of G. For each such F, enumerate all 2^{n-1} vectors $x \in \prod_{a \in A(F)} \{\ell_a, u_a\}$. Interpreting x as a periodic tension defines a periodic timetable $\pi \in [0,T)^{V(G)}$ which can be computed by an $O(n+m)$ depth-first traversal as in the proof of Lemma 7.2.7 (replacing ℓ by x). Use (7.2) to compute the periodic tension x_a of all $O(m)$ remaining co-tree arcs $a \notin A(F)$ and check if the bounds are satisfied. If G is not connected, T-PESP-OPTIMALITY can be solved on each component individually.

3. Let B denote the incidence matrix of G. Then the modulo constraint $B^t\pi \equiv x \bmod T$ is satisfied if and only if there is a vector $p \in \mathbb{Z}^{A(G)}$ such that $B^t\pi = x - Tp$. Since any entry of $B^t\pi$ lies in the interval $(-T, T)$, and any tension computed by (7.2) satisfies $x \in [0, 2T)$, it suffices to consider $p \in \{0,1,2\}^{A(G)}$, cf. C. Liebchen (2006, Lemma 9.2). The algorithm is now to solve the problem

$$\begin{aligned} &\text{Minimize} && w^t y \\ &\text{s.t.} && B^t\pi = x - Tp, \\ & && \ell \le x \le u \end{aligned}$$

for each fixed $p \in \{0,1,2\}^{A(G)}$. This is a series of 3^m linear programs. □

Somewhat unsurprisingly, Lemma 7.2.8 implies that all three presented PESP variants are solvable in polynomial time when the number of events or the number of activities is fixed. In the remainder of the paper, we investigate several graph parameters and their effects on the parameterized complexity of PESP, starting with treewidth.

7.3. PESP on Networks of Treewidth Two

7.3.1. Subset Sum

Definition 7.3.1 (Garey and Johnson, 1979, SP13)**.** The SUBSET SUM problem is the following: Given $r \in \mathbb{N}$, $c \in \mathbb{Z}^r_{\geq 0}$ and $C \in \mathbb{Z}_{\geq 0}$ with $C \leq \sum_{i=1}^r c_i$, is there a $z \in \{0,1\}^r$ such that $c^t z = C$?

The SUBSET SUM problem is weakly NP-complete (Karp, 1972). We will at first construct a polynomial-time reduction of SUBSET SUM to T-PESP-FEASIBILITY.

Definition 7.3.2. Let (r, c, C) be a SUBSET SUM instance as above. Define $I(r, c, C)$ as the instance (G, T, ℓ, u) for T-PESP-FEASIBILITY as depicted in Figure 7.1 with $T := \sum_{i=1}^r c_i + 1$.

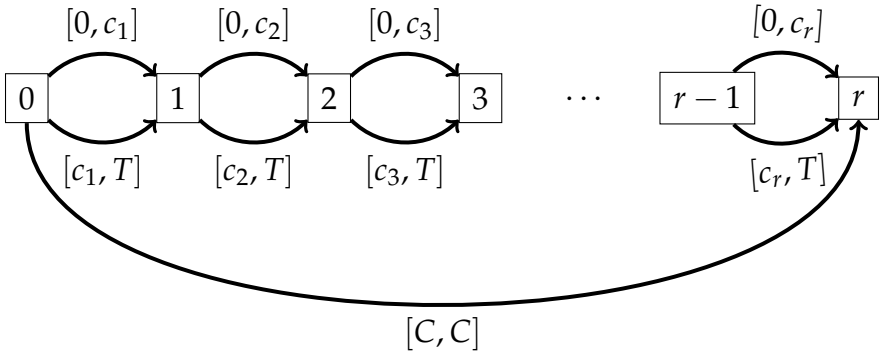


Figure 7.1.: Instance $I(r, c, C)$: arcs a are labeled with $[\ell_a, u_a]$, $T := \sum_{i=1}^r c_i + 1$

Lemma 7.3.3. *The* SUBSET SUM *instance* (r, c, C) *has a solution if and only if* T-PESP-FEASIBILITY *has a solution on the instance* $I(r, c, C)$.

Proof. Let π be a periodic timetable for $I(r, c, C)$. Then $[\pi_i - \pi_{i-1}]_T \in \{0, c_i\}$ holds for all $i \in \{1, \ldots, r\}$. Set

$$z_i := \begin{cases} 1 & \text{if } [\pi_i - \pi_{i-1}]_T = c_i, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, r.$$

For the arc $(0, r)$, we then obtain

$$C = [C]_T = [\pi_r - \pi_0]_T = \left[\sum_{i=1}^r (\pi_i - \pi_{i-1})\right]_T = \left[\sum_{i=1}^r z_i c_i\right]_T = \sum_{i=1}^r z_i c_i,$$

and found a positive answer to the SUBSET SUM problem on (r, c, C). Conversely, any vector $z \in \{0,1\}^r$ such that $c^t z = C$ yields a feasible periodic timetable π by setting

$$\pi_0 := 0 \quad \text{and} \quad \pi_i := \pi_{i-1} + z_i c_i, \quad i = 1, \ldots, r. \qquad \square$$

7.3.2. Treewidth

Definition 7.3.4 (e.g., Neil Robertson and P.D Seymour, 1984)**.** Given a graph G, a *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$ consisting of a tree $\mathcal{T}$ and a family of *bags* $\mathcal{X} = (X_t)_{t \in V(\mathcal{T})}$ with $X_t \subseteq V(G)$ for each $t \in V(\mathcal{T})$ such that

1. $\bigcup_{t \in V(\mathcal{T})} X_t = V(G)$,
2. for each $a \in A(G)$, there is a bag X_t containing both endpoints of a,
3. for each $v \in V(G)$, the subforest of $\mathcal{T}$ induced by $\{t \in V(\mathcal{T}) \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\max_{t \in (\mathcal{T})} |X_t| - 1$, and the *treewidth* of G is defined as the minimum possible width of a tree decomposition, i.e.,

$$\operatorname{tw}(G) := \min \left\{ \max_{t \in V(\mathcal{T})} |X_t| \;\middle|\; (\mathcal{T}, \mathcal{X}) \text{ is a tree decomposition of } G \right\} - 1.$$

This definition applies to both undirected and directed graphs, and as well to multigraphs. According to Definition 7.3.4. the treewidth of a graph with multiple edges equals the treewidth of the graph where all multiple edges between two vertices are replaced by a single edge. The simple connected graphs of treewidth 1 are precisely the trees.

Lemma 7.3.5. *For any* SUBSET SUM *instance* (r, c, C) *with* $r \geq 2$*, the event-activity network* G *of the instance* $I(r, c, C)$ *has treewidth 2.*

Proof. The path of length r depicted in Figure 7.2, where each node is labeled with its bag, is a tree decomposition of G. Checking the properties of a tree decomposition is

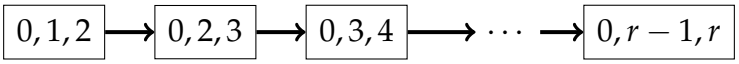


Figure 7.2.: An optimal tree decomposition of width 2 for the $I(r, c, C)$ network

straightforward. The maximum bag size is 3, and hence $\operatorname{tw}(G) \leq 2$. Removing one of the two arcs from i to $i + 1$, $i = 0, \ldots, r - 1$, does not change the treewidth in the sense of Definition 7.3.4. As $r \geq 2$, this results in a simple graph containing a cycle, so that $\operatorname{tw}(G) \geq 2$. □

An alternative way to see that the network G of $I(r, c, C)$ has treewidth at most 2 is to observe that G is series-parallel (see, e.g., Bodlaender and Antwerpen-de Fluiter, 2001, Lemma 3.4). Combining Lemma 7.3.3 and Lemma 7.3.5, we obtain:

Theorem 7.3.6. *T-*PESP-FEASIBILITY *is NP-complete on networks of treewidth at most 2.*

Since the transformation in the proof of Theorem 7.2.6 reduces the T-PESP-FEASIBILITY problem to T-RPESP-OPTIMALITY by adding antiparallel arcs, which does not alter the treewidth, we have moreover:

Theorem 7.3.7. *T-RPESP-OPTIMALITY is NP-complete on networks of treewidth at most 2.*

Remark 7.3.8. If simple graphs are desired, one can dispose of the parallel arcs in $I(r, c, C)$ by subdividing any arc with bounds $[0, c_i]$ into two arcs with bounds $[0, c_i]$ and $[0, 0]$ without affecting the feasibility of the PESP instance. Moreover, one checks that this does not increase the treewidth of $I(r, c, C)$.

Remark 7.3.9. In the proof of Lemma 7.3.3, the period time T is chosen very large. The above NP-completeness theorems hence do not hold when T is fixed. We will give a pseudo-polynomial algorithm for T-PESP-OPTIMALITY with bounded treewidth in Section 7.4, showing that fixing both T and the treewidth results in a polynomial-time algorithm.

Remark 7.3.10. We want to remark that there has already been a paper (Heuven van Staereling, 2018) titled *Tree Decomposition Methods for the Periodic Event Scheduling Problem*. However, the title is misleading, as the algorithm there is about finding trees *in* the network rather than considering tree decompositions in the usual sense.

7.3.3. Branchwidth

Definition 7.3.11 (Neil Robertson and P.D Seymour, 1991)**.** Given a graph G, a *branch decomposition* of G is a pair $(\mathcal{B}, \varphi)$, where $\mathcal{B}$ is a tree such that every non-leaf node has degree 3, and φ is a bijection from the leaves of $\mathcal{B}$ to $A(G)$. Deleting an edge e of $\mathcal{B}$ disconnects $\mathcal{B}$ into two subtrees and hence partitions the leaves of $\mathcal{B}$ into two sets. Applying φ, this yields a partition $A = A_e^1 \dot\cup A_e^2$. This defines in turn a *vertex separator* $S_e \subseteq V$ as the set of vertices that are incident both to an edge in A_e^1 and to an edge in A_e^2.
The *width* of a branch decomposition $(\mathcal{B}, \varphi)$ is defined as $\max_{e \in E(\mathcal{B})} |S_e|$. The *branchwidth* of G is then the minimum possible width of a branch decomposition, i.e.,

$$\mathrm{bw}(G) := \min \left\{ \max_{e \in E(\mathcal{B})} |S_e| \;\middle|\; (\mathcal{B}, \varphi) \text{ is a branch decomposition of } G \right\}.$$

Treewidth and branchwidth are related as follows:

Theorem 7.3.12 (Neil Robertson and P.D Seymour, 1991, 5.1)**.** *If* $\mathrm{bw}(G) \geq 2$*, then*

$$\mathrm{bw}(G) \leq \mathrm{tw}(G) + 1 \leq \left\lfloor \frac{3}{2} \mathrm{bw}(G) \right\rfloor.$$

It follows immediately that the problems T-PESP-FEASIBILITY and T-RPESP-OPTIMALITY are NP-complete on networks with branchwidth 3. We prove below that the NP-completeness is already given for branchwidth 2.

Lemma 7.3.13. *For any* SUBSET SUM *instance (r, c, C), the event-activity network G of the instance $I(r, c, C)$ has branchwidth 2.*

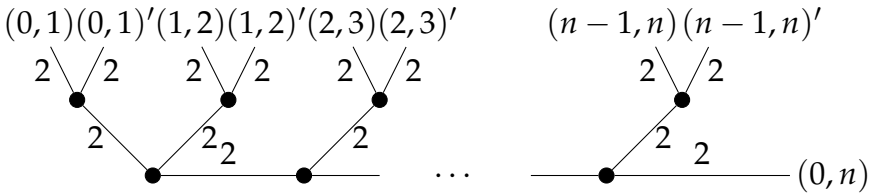


Figure 7.3.: An optimal branch decomposition of width 2 for the $I(r, c, C)$ network

Proof. Figure 7.3 shows a branch decomposition of G, where the leaves are labeled with the corresponding arc, and the edges are labeled with the cardinality of the corresponding vertex separator. This is clearly a branch decomposition. Checking the cardinalities of the vertex separators is again straightforward, and hence $\mathrm{bw}(G) \leq 2$. The network G cannot have branchwidth 1, as in any branch decomposition, the edge incident to the leaf representing $(0, r)$ always induces the vertex separator $\{0, r\}$ of size 2. □

Theorem 7.3.14. *T-PESP-Feasibility and T-RPESP-Optimality are NP-complete on networks with branchwidth at most 2.*

Proof. For T-PESP-Feasibility, this follows from Lemma 7.3.3 and Lemma 7.3.13. Introducing anti-parallel arcs in the network of $I(r, c, C)$ does not increase the branchwidth of 2: In the branch decomposition of the proof of Lemma 7.3.13, replace a leaf with a vertex of degree 3 adjacent to two new leaves corresponding to the two anti-parallel arcs. The new edges have vertex separators of size 2. Consequently, the transformation in the proof of Theorem 7.2.6 does not alter the branchwidth, and T-RPESP-Optimality is NP-complete on networks with branchwidth at most 2. □

Another approach to prove Lemma 7.3.13 and Theorem 7.3.14 is to exploit that a graph of treewidth at most 2 has branchwidth at most 2 (combine, e.g., Bodlaender and Antwerpen-de Fluiter, 2001, Lemma 3.5 with Neil Robertson and P.D Seymour, 1991, 4.2).

7.3.4. Carvingwidth

Carvingwidth is defined analogously to branchwidth, by labeling the leaves of an unrooted binary tree with the vertices of the original graph instead of the edges.

Definition 7.3.15 (P. D. Seymour and Thomas, 1994)**.** Given a graph G, a *carving decomposition* of G is a pair $(\mathcal{C}, \psi)$, where $\mathcal{C}$ is a tree such that every non-leaf node has degree 3, and ψ is a bijection from the leaves of $\mathcal{C}$ to $V(G)$. Removing an edge e of $\mathcal{C}$ induces a partition of the leaves of $\mathcal{C}$, and hence via ψ also a partition $V(G) = V_e^1 \dot\cup V_e^2$. Let $\delta(V_e^1)$ $(= \delta(V_e^2))$ denote the set of cut edges.
The maximum cardinality of $\delta(V_e^1)$ taken over all $e \in E(\mathcal{C})$ is the *width* of $(\mathcal{C}, \psi)$. The *carvingwidth* of G is defined as

$$\mathrm{cw}(G) := \min \left\{ \max_{e \in E(\mathcal{C})} |\delta(V_e^1)| \;\middle|\; (\mathcal{C}, \psi) \text{ is a carving decomposition of } G \right\}.$$

The definition of carvingwidth applies to multigraphs as well, but in contrast to branch- and treewidth, it is sensitive to multiple edges: The carvingwidth is at least the maximum vertex degree $\Delta(G)$, as $\mathrm{cw}(G) \geq \deg(v) = |\delta(v)|$ for each $v \in V(G)$. Carvingwidth is related to branchwidth as follows:

Theorem 7.3.16 (Nestoridis and Thilikos, 2014; Eppstein, 2018)**.** *Let G be a graph with maximum vertex degree $\Delta(G)$. Then*

$$\max\left(\Delta(G), \left\lceil \frac{1}{2}\mathrm{bw}(G) \right\rceil\right) \leq \mathrm{cw}(G) \leq \Delta(G) \cdot \mathrm{bw}(G).$$

Theorem 7.3.17. *T-PESP-Feasibility is NP-complete on networks of carvingwidth at most 3, and T-RPESP-Optimality is NP-complete on networks of carvingwidth at most 6.*

Proof. Let $r \geq 2$ and consider again an instance of the form $I(r, c, C)$ with event-activity network G. Then $\Delta(G) = 4$, so that $\mathrm{cw}(G) \geq 4$. For all $i \in \{1, \ldots, r-1\}$, split vertex i into two new vertices i^+ and i^-, connected by a single directed arc (i^+, i^-) with bounds $\ell_{i^+,i^-} = u_{i^+,i^-} = 0$. The splitting is done in such a way that the arcs entering i are now entering i^+, and the arcs leaving i are leaving i^-. Any periodic timetable has the same value at i^+ and i^-, so that the proof of Lemma 7.3.3 carries over. However, the modified graph has maximum degree 3. The carving decomposition in Figure 7.4, where the edges labeled with the number of cut edges, has width 3. Hence T-PESP-Feasibility

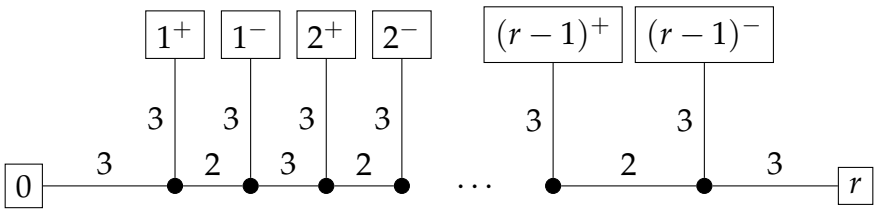


Figure 7.4.: An optimal carving decomposition of width 3 of the modified $I(r, c, C)$ network

is NP-complete on networks of carvingwidth 3. As a consequence, keeping in mind the arc duplication occurring in the proof of Theorem 7.2.6, T-RPESP-Optimality is NP-complete for carvingwidth 6. □

Remark 7.3.18. *T-PESP-Optimality is trivial to solve on graphs G with $\mathrm{cw}(G) = 1$, as then $\Delta(G) \leq 1$ by Theorem 7.3.16. If $\mathrm{cw}(G) = 2$, then $\Delta(G) \leq 2$, so that any weakly connected component of G, seen as an undirected graph, is either a path or a cycle. T-PESP-Optimality is solvable in linear time on paths (Lemma 7.2.7). We will see later in Theorem 7.5.6 that T-PESP-Optimality admits a polynomial-time algorithm for bounded cyclomatic number, and in particular on a single cycle.*

7.4. Dynamic Programs

The SUBSET SUM problem is weakly NP-complete and can be solved by pseudo- polynomial-time algorithms. In the following, we present two dynamic programs for T-PESP-OPTIMALITY running in pseudo- polynomial time for event-activity networks of bounded treewidth and branchwidth, respectively. Since T-PESP-OPTIMALITY comprises both T-PESP-FEASIBILITY and T-RPESP-OPTIMALITY, this implicitly gives pseudo-polynomial time algorithms for these problems as well.

7.4.1. PESP and Vertex Separators

The key insight for our dynamic programming approach is the following decomposition property: Let $I = (G, T, \ell, u, w)$ be a T-PESP-OPTIMALITY instance. For any partition $A(G) = A^1 \dot\cup A^2$, we can partition I into two subinstances I^1 resp. I^2 restricted to the activities in A^1 resp. A^2. If y is a feasible periodic slack on I, then the restrictions y^1 resp. y^2 to I^1 resp. I^2 yield feasible periodic slacks with the property

$$\sum_{a \in A(G)} w_a y_a = \sum_{a \in A^1} w_a y^1_a + \sum_{a \in A^2} w_a y^2_a. \tag{7.3}$$

On the level of timetables, we obtain from a timetable π on I two timetables π^1 resp. π^2 on I^1 resp. I^2 such that π, π^1 and π^2 all coincide when restricted to the events of the vertex separator S associated to the partition of $A(G)$ as in Definition 7.3.11.

Conversely, we can glue two periodic timetables π^1 and π^2 together to a timetable π on I if the restrictions to a vertex separator S satisfy $\pi^1|_S = \pi^2|_S$. For the corresponding periodic slacks then holds Equation (7.3).

Definition 7.4.1. Let $I = (G, T, \ell, u, w)$ be a T-PESP-OPTIMALITY instance, and let $S \subseteq V(G)$. For a vector $\rho \in [0, T)^S$, define $\mathrm{OPT}(I, S, \rho)$ as the minimum weighted slack of a periodic timetable π on I when additionally $\pi|_S = \rho$ is required.

For minimum weighted slacks, the above discussion shows the following:

Lemma 7.4.2. *Let $I = (G, T, \ell, u, w)$ be a feasible T-PESP-OPTIMALITY instance. Let $A(G) = A^1 \dot\cup A^2$ be a partition with vertex separator S giving subinstances I^1 and I^2 of I. Then*

$$\mathrm{OPT}(I) = \min\{\mathrm{OPT}(I^1, S, \pi|_S) + \mathrm{OPT}(I^2, S, \pi|_S) \mid \pi \text{ is a feasible periodic timetable on } I\}.$$

More generally, if additionally the timetable on $W \subseteq V(G)$ is fixed to $\rho \in [0, T)^W$, then

$$\mathrm{OPT}(I, W, \rho) = \min\{\, \mathrm{OPT}(I^1, S \cup W^1, \pi|_{S \cup W^1}) + \mathrm{OPT}(I^2, S \cup W^2, \pi|_{S \cup W^2}) \mid \pi \text{ is a feasible periodic timetable on } I \text{ with } \pi|_W = \rho\},$$

where $W^i = W \cap V^i$ denotes the intersection of W with the set of events V^i of I^i, $i = 1, 2$.

7.4.2. A Branch Decomposition Approach

Since branch decompositions naturally encode vertex separators, we describe at first a branch-decomposition-based dynamic program for T-PESP-OPTIMALITY. Let $I = (G, T, \ell, u, w)$ be a T-PESP-OPTIMALITY instance. We assume that G is 2-edge-connected when seen as undirected graph. Let $(\mathcal{B}, \varphi)$ be a branch decomposition of G with node set $V(\mathcal{B})$ and edge set $E(\mathcal{B})$. Subdivide an arbitrary edge of $E(\mathcal{B})$ and call the new node τ the *root*. Recall from Definition 7.3.11 that every edge $e \in E(\mathcal{B})$ corresponds to a partition $A = A_e^1 \dot\cup A_e^2$ with vertex separator S_e. We assume that A_e^1 is the subset of activities coming from the component of $\mathcal{B} \setminus \{e\}$ not containing the root τ.

Algorithm 7.4.3. For each edge $e \in E(\mathcal{B})$, we compute an $|S_e|$-dimensional table F_e having an entry for each $\pi \in \{0, \dots, T-1\}^{S_e}$. The table F_e is filled by a dynamic program starting from the edges $e \in E(\mathcal{B})$ incident to leaves and with decreasing distance to the root:

1. If $e \in E(\mathcal{B})$ is incident to a leaf corresponding via φ to an activity $ij \in A(G)$, then set
$$F_e(\pi) := \begin{cases} w_{ij}[\pi_j - \pi_i - \ell_{ij}]_T & \text{if } [\pi_j - \pi_i - \ell_{ij}]_T \le u_{ij} - \ell_{ij}, \\ \infty & \text{otherwise.} \end{cases}$$
2. If e is incident to two edges e_1, e_2 with larger distance from the root, then set
$$F_e(\pi) := \min\{F_{e_1}(\pi'|_{S_{e_1}}) + F_{e_2}(\pi'|_{S_{e_2}}) \mid \pi' \in \{0, \dots, T-1\}^{S_{e_1} \cup S_{e_2}}, \pi'|_{S_e} = \pi\}.$$
3. If the tables of the two edges e_1, e_2 incident to τ have been computed, return
$$\min\{F_{e_1}(\pi) + F_{e_2}(\pi) \mid \pi \in \{0, \dots, T-1\}^{S_{e_1}}\}.$$

Lemma 7.4.4. *Let $e \in E(\mathcal{B})$, $\pi \in \{0, \dots, T-1\}^{S_e}$. Denote by I_e the subinstance of I containing precisely the activities in A_e^1.*

1. *If $F_e(\pi) < \infty$, then $F_e(\pi) = \mathrm{OPT}(I_e, S_e, \pi)$.*
2. *If $F_e(\pi) = \infty$, then I_e is infeasible.*

Proof. Recall from Section 7.2 that it suffices to consider timetables with values in the discrete set $\{0, \dots, T-1\}$, as ℓ and u are integer.

If e is incident to a leaf associated to an activity $ij \in A(G)$, then $A_e^1 = \{ij\}$ and $S_e = \{i, j\}$, as G is 2-edge-connected. Hence $\mathrm{OPT}(I_e, S_e, \pi)$ is the minimum weighted slack of the activity ij when the timetable at i resp. j is fixed to π_i resp. π_j. Therefore we set $F_e(\pi)$ to $w_{ij}[\pi_j - \pi_i - \ell_{ij}]_T$ if the slack $[\pi_j - \pi_i - \ell_{ij}]_T$ is feasible and otherwise to ∞.

Otherwise, let e be adjacent to $e_1, e_2 \in E$, with e_1, e_2 having larger distance from the root than e. Then $A_e^1 = A_{e_1}^1 \dot\cup A_{e_2}^1$ and hence $S_e \subseteq S_{e_1} \cup S_{e_2}$. Moreover, $(S_{e_1} \cup S_{e_2}) \setminus S_e$ is the vertex separator of the partition of $A_e^1 = A_{e_1}^1 \dot\cup A_{e_2}^1$. Applying Lemma 7.4.2 for $W = S_e$ and $S = (S_{e_1} \cup S_{e_2}) \setminus S_e$ yields the formula in Algorithm 7.4.3. □

Lemma 7.4.5. *If $k = \max_{e \in E(\mathcal{B})} |S_e|$ and $m = |A(G)|$, then Algorithm 7.4.3 computes $\mathrm{OPT}(I)$ or decides that I is infeasible in $O(mT^{\lfloor 3k/2 \rfloor})$ time.*

Proof. We first consider correctness. At the root τ with incident edges e_1, e_2, Algorithm 7.4.3 computes the tables F_{e_1}, F_{e_2} such that $F_{e_i}(\pi) = \mathrm{OPT}(I_{e_i}, S_{e_i}, \pi)$ for all π and $i = 1, 2$ by Lemma 7.4.4. Observe that $A^1_{e_1} = A^2_{e_2}$, $A^2_{e_1} = A^1_{e_2}$, and $S_{e_1} = S_{e_2}$, so that by the first equation in Lemma 7.4.2, $\mathrm{OPT}(I) = \min\{\mathrm{OPT}(I_{e_1}, S_{e_1}, \pi) + \mathrm{OPT}(I_{e_2}, S_{e_2}, \pi) \mid \pi \in \{0, \ldots, T-1\}^V \text{ feasible timetable}\}$. This is precisely reflected in the third step of Algorithm 7.4.3, treating infeasible subinstances with infinite objective value.

Concerning running time, Step 1 can be done in $O(T^2)$ time and is called m times. Step 3 takes $O(T^k)$ time since $|S_{e_1}| \leq k$. As any node in the rooted branch decomposition has either 0 or 2 children and there are m leaves, there are $m - 1$ edges for which Step 2 is called. Each of the $|S_e|$ table entries requires to take a minimum over $|(S_{e_1} \cup S_{e_2}) \setminus S_e|$ previously computed table entries.

We claim that

$$2|S_{e_1} \cup S_{e_2}| \leq |S_{e_1}| + |S_{e_2}| + |S_e|.$$

If $i \in S_{e_1} \setminus S_{e_2}$, then i is adjacent to an activity $a \in A^1_{e_1}$ and $a' \notin A^1_{e_1}$. Since $A^1_{e_1}$ and $A^1_{e_2}$ are disjoint, $a \notin A^1_{e_2}$. As $i \notin S_{e_2}$, then also $a' \notin A^1_{e_2}$, and consequently $a' \notin A^1_e = A^1_{e_1} \cup A^1_{e_2}$. It follows that $i \in S_e$, and any such i appears hence twice in the right-hand side of the above inequality. By symmetry, the same holds for all $i \in S_{e_2} \setminus S_{e_1}$. Clearly, any $i \in S_{e_1} \cap S_{e_2}$ is counted both in $|S_{e_1}|$ and $|S_{e_2}|$. This proves the claim. The relation between the separators is also depicted in Figure 7.5.

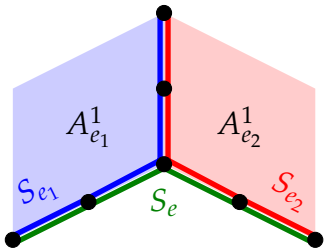


Figure 7.5.: Relation between S_e, S_{e_1} and S_{e_2}

Since the size of any of the three vertex separators is bounded by k, we obtain

$$|S_e| + |(S_{e_1} \cup S_{e_2}) \setminus S_e| = |S_{e_1} \cup S_{e_2}| \leq \left\lfloor \frac{3k}{2} \right\rfloor.$$

Thus, Step 2 accounts in total for a running time of $O(mT^{\lfloor 3k/2 \rfloor})$, and this dominates the other steps, as $k \geq 2$ since $|S_e| = 2$ for every edge e incident to a leaf of $\mathcal{B}$. □

Having presented the core dynamic program, we now turn to the surrounding problems:

Finding an optimal branch decomposition. If $\mathrm{bw}(G) \leq k$, then there is a linear-time algorithm computing a branch decomposition of width $\leq k$ (Bodlaender and Thilikos, 1997).

Computing an optimal timetable. By additional bookkeeping, we can not only compute the minimum weighted slack, but also a periodic timetable realizing this slack.

2-edge-connectedness. It is clear from the description of T-PESP-OPTIMALITY that the problem can be solved on each weakly connected component of G individually. Moreover, if one of these components is not 2-edge-connected, then the optimal periodic slack of any bridge will be zero (Borndörfer et al., 2019, §3.2). Hence one can safely assume w.l.o.g. that G is 2-edge-connected. Note that this assumption implies $\mathrm{bw}(G) \geq 2$.

Fixing. If π is a feasible periodic timetable and $d \in \mathbb{R}$, then the timetable π' defined by $\pi'_i := [\pi_i + d]_T$ for all $i \in V(G)$ is feasible as well and produces the same periodic slack. In particular, in Algorithm 7.4.3, for all $e \in E$, one can choose an event $i \in S_e$ and then fix $\pi_i = 0$. Thus Algorithm 7.4.3 can be adapted to run in $O(mT^{\lfloor 3k/2 \rfloor - 1})$ time.

As a consequence, in conjunction with Lemma 7.2.7 and Theorem 7.3.14, we obtain:

Theorem 7.4.6. *For $k \in \mathbb{N}$, there is an $O(mT^{\lfloor 3k/2 \rfloor - 1})$ algorithm solving T-PESP-OPTIMALITY on event-activity networks G with m activities and $\mathrm{bw}(G) \leq k$. In particular, if $k \geq 2$ is fixed, then T-PESP-OPTIMALITY, T-PESP-FEASIBILITY and T-RPESP-OPTIMALITY are all weakly NP-complete.*

7.4.3. A Tree Decomposition Version

In this subsection, we develop a tree decomposition analogue of Algorithm 7.4.3. Let $I = (G, T, \ell, u, w)$ be a T-PESP-OPTIMALITY instance. Let $(\mathcal{T}, \mathcal{X})$ be a tree decomposition. We assume that $(\mathcal{T}, \mathcal{X})$ is *rooted*, i.e., we pick an arbitrary leaf node $\tau \in V(\mathcal{T})$ and turn $\mathcal{T}$ into an arborescence where all edges point away from τ. Further suppose that the tree decomposition is *nice* (Kloks, 1994, Definition 13.1.15), i.e., each node $t \in V(\mathcal{T})$ with bag X_t fits into precisely one of the following categories:

- *Root*: $t = \tau$,
- *Leaf*: $t \neq \tau$, t has no children, and $|X_t| = 1$,
- *Introduce*: t has exactly one child u and one parent, $X_u \subseteq X_t$, and $|X_t| = |X_u| + 1$,
- *Forget*: t has exactly one child u and one parent, $X_t \subseteq X_u$, and $|X_t| = |X_u| - 1$,
- *Join*: t has exactly two children u_1, u_2 and one parent, and $X_t = X_{u_1} = X_{u_2}$.

Algorithm 7.4.7. For each node $t \in V(\mathcal{T})$, we compute a $|X_t|$-dimensional table D_t having an entry for each $\pi \in \{0, \dots, T-1\}^{X_t}$. The table D_t is filled by the following dynamic program from the leaves to the root, depending on the role of the node t in the nice tree decomposition:

- If $t \neq \tau$ is a *leaf*, then $X_t = \{i\}$ for a single event $i \in V$. For all $\pi \in \{0, \dots, T-1\}$, set $D_t(\pi) := 0$.
- If t is an *introduce* vertex with child u, then $X_t = X_u \cup \{i\}$ for some event i. Put

$$D_t(\pi) := D_u(\pi|_{X_u}) + \sum_{ij \in A(G):\, j \in X_u} w_{ij} y_{ij} + \sum_{ji \in A(G):\, j \in X_u} w_{ji} y_{ji},$$

  where, for $ij \in A(G)$,

$$y_{ij} := \begin{cases} [\pi_j - \pi_i - \ell_{ij}]_T & \text{if } [\pi_j - \pi_i - \ell_{ij}]_T \leq u_{ij} - \ell_{ij}, \\ \infty & \text{otherwise.} \end{cases}$$

- If t is a *forget* vertex with child u, then $X_t = X_u \setminus \{i\}$ for some event i. Set

$$D_t(\pi) := \min\{D_u(\pi') \mid \pi' \in \{0, \dots, T-1\}^{X_u}, \pi'|_{X_t} = \pi\}.$$

- If t is a *join* vertex with children u_1 and u_2, then $X_t = X_{u_1} = X_{u_2}$. If $D_{u_1}(\pi) < \infty$ and $D_{u_2}(\pi) < \infty$, define

$$D_t(\pi) := D_{u_1}(\pi) + D_{u_2}(\pi) - \sum_{ij \in A:\, i \in X_t, j \in X_t} w_{ij}[\pi_j - \pi_i - \ell_{ij}]_T,$$

  otherwise set $D_t(\pi) := \infty$.
- If $t = \tau$ is the *root*, then compute D_τ treating τ as a forget node. Return

$$\min\{D_\tau(\pi) \mid \pi \in \{0, \dots, T-1\}^{X_\tau}\}.$$

For a node $t \in V(\mathcal{T})$, define G_t as the subnetwork of G induced by the events in the bag of X_t and the bags of all descendants of t in $\mathcal{T}$. Denote by I_t the corresponding T-PESP-OPTIMALITY subinstance of I.

Lemma 7.4.8. *Let $t \in V(\mathcal{T})$ be a node of the tree decomposition and $\pi \in \{0, \dots, T-1\}^{X_t}$.*

1. *If $S_t(\pi) < \infty$, then $S_t(\pi) = \mathrm{OPT}(I_t, X_t, \pi)$.*
2. *If $S_t(\pi) = \infty$, then there is no feasible peridoic timetable on I_t coinciding with π on X_t.*

Proof. Let $G_t = (V_t, A_t)$. If $t \neq \tau$ is a leaf, then $A_t = \emptyset$, and both statements are trivial.
Introducing an event i at t with child u means that

$$V_t = V_u \,\dot\cup\, \{i\}, \quad A_t = A_u \,\dot\cup\, (\{ij \in A : j \in X_u\} \,\dot\cup\, \{ji \in A : j \in X_u\}).$$

The latter is a partition of the activities of G_t whose vertex separator is contained in X_u. By Lemma 7.4.2, $D_t(\pi)$ must hence equal $D_u(\pi|_{X_u})$ plus the minimum weighted slack of the subinstance associated to $A_t \setminus A_u$ fixing the timetable at X_t, and the latter is precisely given by the formula in Algorithm 7.4.7.

When forgetting an event i at t with child u, then $G_t = G_u$. However, the timetable has to be fixed at X_t which containes one vertex less than X_u, so that we minimize over all table entries where the timetable restricted to X_u is the same.

If t is a join vertex with u_1 and u_2 as children, then $G_t = G_{u_1} \cup G_{u_2}$. We apply Lemma 7.4.2 to $A_t = A_{u_1} \dot\cup (A_{u_2} \setminus A_{u_1})$ and to $A_{u_2} = (A_{u_2} \setminus A_{u_1}) \dot\cup (A_{u_1} \cap A_{u_2})$. Note that by the connectedness property of the tree decomposition, any activity $A_{u_1} \cap A_{u_2}$ has both endpoints in $X_t = X_{u_1} = X_{u_2}$, so that X_t contains the vertex separators of both partitions. Now the minimum weighted slack on G_t is the sum of the minimum weighted slacks on G_{u_1} and G_{u_2}, subtracting the minimum weighted slack of the activities that have been counted twice, i.e., $A_{u_1} \cap A_{u_2}$. Hence the minimum weighted slack on G_t can be computed as described in Algorithm 7.4.7. If fixing the timetable π on X_t yields an infeasible timetable, then either $D_{u_1}(\pi)$ or $D_{u_2}(\pi)$ must have been infinite, and vice versa.

Finally, if $t = \tau$ is the root, then τ has degree one and hence has a unique child u, so that we can treat it as a forget node. □

Lemma 7.4.9. *If I is feasible, then Algorithm 7.4.7 returns $\mathrm{OPT}(I)$, and otherwise ∞. Moreover, Algorithm 7.4.7 runs in $O(|V(\mathcal{T})|T^{k+1})$ time, where $k := \max_{t \in V(\mathcal{T})} |X_t| - 1$.*

Proof. By Lemma 7.4.8, at the root τ, the table entry $D_\tau(\pi)$ is the minimum weighted slack on the subnetwork $G_\tau = G$ fixing the timetable at X_τ to π (or ∞). Minimizing over all entries in D_τ hence gives the minimum weighted slack of the T-PESP-Optimality instance (or detects infeasibility). The running time estimate is straightforward, as we need to fill each of the $|V(\mathcal{T})|$ tables with at most T^{k+1} entries and only employ summation and minimization. □

Theorem 7.4.10. *For $k \in \mathbb{N}$, there is an $O(nT^k)$ algorithm solving T-PESP-Optimality on event-activity networks G with n events and $\mathrm{tw}(G) \leq k$.*

Proof. By Bodlaender (1996), if $\mathrm{tw}(G) \leq k$, then a tree decomposition with $O(n)$ nodes realizing width $\mathrm{tw}(G)$ can be found in $O(f(k) \cdot n)$ time. This can be transformed into a nice tree decomposition on $O(n)$ nodes within $O(n)$ time (Kloks, 1994, Lemma 13.1.3). Applying Lemma 7.4.9 provides an $O(nT^{k+1})$ algorithm. Using the same fixing strategy as for Theorem 7.4.6, we obtain an $O(nT^k)$ algorithm. □

Remark 7.4.11. The branch-decomposition based algorithm of Theorem 7.4.6 has a running time of $O(mT^{\lfloor 3\,\mathrm{bw}(G)/2 \rfloor - 1})$ time, and the tree-decomposition based one runs in $O(nT^{\mathrm{tw}(G)})$, see Theorem 7.4.10. By Theorem 7.3.12, if G is 2-edge-connected, then $nT^{\mathrm{tw}(G)} \leq mT^{\lfloor 3\,\mathrm{bw}(G)/2 \rfloor - 1}$, so that the tree-decomposition-based algorithm is expected to be asymptotically superior.

Remark 7.4.12. In terms of memory, Algorithm 7.4.3 with the fixing strategy needs to store at most $T^{\mathrm{bw}(G)-1}$ table entries per edge, whereas Algorithm 7.4.7 (with fixing) stores at most $T^{\mathrm{tw}(G)-1}$ entries per node. In both cases, at most 3 tables need to be stored at the same time, as any node in one of the decompositions has at most 2 children. Since $\mathrm{bw}(G) - 1 \leq \mathrm{tw}(G)$ if $\mathrm{bw}(G) \geq 2$, the branch-decomposition-based method potentially requires less space.

Remark 7.4.13. We omit a carving-decomposition-based algorithm. Bounding the carvingwidth by k means that the branchwidth is bounded by $2k$ according to Theorem 7.3.16 and we can invoke Algorithm 7.4.3.

7.5. Fixed-parameter tractable algorithms

We have already seen in Lemma 7.2.8 that fixing the number of events or the number of activities leads to fixed-parameter tractable algorithms for T-PESP-OPTIMALITY. In this section, we discuss fixed-parameter tractability by cyclomatic number, diameter, and vertex cover number. The cyclomatic number is a common measure for the difficulty of PESP instances, as it counts the number of integral variables in a cycle-based mixed integer programming formulation (Borndörfer et al., 2019). The size of a minimum vertex cover has led to fixed-parameter algorithms for several coloring problems where bounding the treewidth does still result in NP-hard problems (Fiala et al., 2011), as it is the case for the PESP family.

7.5.1. Cyclomatic Number

Definition 7.5.1. Let G be a graph on n vertices, m edges and c weakly connected components. The *cyclomatic number* of G is defined as $\mu(G) := m - n + c$.

Alternatively, the cyclomatic number is the dimension of the cycle space of G.

Lemma 7.5.2. *Let G be a graph. Then* $\operatorname{tw}(G) \leq \mu(G) + 1$.

Proof. We give a proof by induction on $\mu(G)$. If $\mu(G) = 0$, then G is a forest and therefore $\operatorname{tw}(G) = 1$.

Now let G be a graph with $\mu(G) > 0$. Then G contains a cycle and hence an edge e such that $G' := G \setminus \{e\}$ has the same number of connected components as G, and $\mu(G') = \mu(G) - 1$. By induction hypothesis, $\operatorname{tw}(G') \leq \mu(G') + 1 = \mu(G)$, so that we find a tree decomposition $(\mathcal{T}, \mathcal{X})$ of G' with maximum bag size at most $\mu(G) + 1$. If there is a bag containing both endpoints of e, then $(\mathcal{T}, \mathcal{X})$ is also a valid tree decomposition for G, and so $\operatorname{tw}(G) \leq \mu(G)$. Otherwise let i be an endpoint of e and add i to each bag of $(\mathcal{T}, \mathcal{X})$. This is a tree decomposition for G of width $\mu(G) + 1$, so that $\operatorname{tw}(G) \leq \mu(G) + 1$. □

While it is true that treewidth and branchwidth can be bounded in terms of each other (see Theorem 7.3.12), this does not hold for treewidth and cyclomatic number:

Lemma 7.5.3. *For $k \geq 2$, there is a class $\mathcal{C}_k$ of simple connected graphs such that* $\operatorname{tw}(G) \leq k$ *holds for all $G \in \mathcal{C}_k$, but for any $N \in \mathbb{N}$, there is a graph $G \in \mathcal{C}$ with $\mu(G) \geq N$.*

Proof. Let $\mathcal{C}$ be the class of graphs G built from a finite disjoint union of cliques of size k with vertex sets $V_1, \ldots, V_r$ together with one additional vertex v joined to each vertex from each clique. Let $\mathcal{T}$ be a path on the vertices $\{1, \ldots, r\}$, and set $X_i := V_i \cup \{v\}$,

$i = 1, \dots, r$. Then $(\mathcal{T}, \mathcal{X})$ is a tree decomposition of G and, as $|X_i| = k+1$, we have $\mathrm{tw}(G) \leq k$. The cyclomatic number of G is given by

$$\mu(G) = \left(r \cdot \frac{k(k-1)}{2} + rk \right) - (rk+1) + 1 = r \cdot \frac{k(k-1)}{2},$$

and for $k \geq 2$, this goes to infinity as $r \to \infty$. □

Theorem 7.5.4. *On networks where no vertex has degree 2, T-PESP-OPTIMALITY is fixed-parameter tractable when parameterized by the cyclomatic number.*

Proof. Let (G, T, ℓ, u, w) be a T-PESP-OPTIMALITY instance. We can safely remove all $i \in V(G)$ with $\deg(i) = 1$, as in any optimal solution, the incident activity must have periodic slack 0. Hence we can assume that G has minimum degree 3. By the Handshaking Lemma,

$$2m = \sum_{i \in V(G)} \deg(i) \geq 3n,$$

and hence

$$\mu = m - n + c \geq \frac{n}{2} + 1,$$

so that $n \leq 2\mu - 2$. This means that fixing μ provides a fixed bound on the number n of events, and we conclude by Lemma 7.2.8. □

Corollary 7.5.5. *T-PESP-FEASIBILITY is fixed-parameter tractable when parameterized by the cyclomatic number.*

Proof. Let (G, T, ℓ, u) be a T-PESP-FEASIBILITY instance. Remove all events of degree 1 from G, as this does neither affect feasibility nor alter the cyclomatic number. Now all degree 2 vertices of G are arranged on (undirected) paths between two vertices of degree ≥ 3. Consider such a path from s to t with $\deg(s), \deg(t) \geq 3$, forward activities $a_1, \dots, a_r$ and backward activities $b_1, \dots, b_s$. Delete all intermediate vertices between s and t and insert a single activity a from s to t with

$$\ell_a := \sum_{i=1}^{r} \ell_{a_i} - \sum_{j=1}^{s} u_{b_j} \quad \text{and} \quad u_a := \sum_{i=1}^{r} u_{a_i} - \sum_{j=1}^{s} \ell_{b_j}.$$

Clearly, if x is a feasible tension with $\ell_{a_i} \leq x \leq u_{a_i}$ and $\ell_{b_j} \leq x_{b_j} \leq u_{b_j}$ for all i and j, then also $\ell_a \leq x_a \leq u_a$ with $x_a := \sum_{i=1}^{r} x_{a_i} - \sum_{j=1}^{s} x_{b_j}$. Conversely, any x_a with $\ell_a \leq x_a \leq u_a$ can be split into feasible tensions on all a_i and b_j. Thus, this transformation preserves feasibility. Moreover, contracting vertices of degree 2 does not change the cyclomatic number, so that we can assume that G has minimum degree 3. Invoke Theorem 7.5.4. □

Theorem 7.5.6. *For fixed cyclomatic number μ, T-PESP-OPTIMALITY is polynomial-time solvable.*

Proof. Let F be a spanning forest of G and let $\gamma_1, \ldots, \gamma_\mu$ be its fundamental cycles, seen as incidence vectors $\{-1, 0, 1\}^{A(G)}$. Then (e.g., Nachtigall, 1998) $x \in \mathbb{R}^{A(G)}$ is a feasible periodic tension if and only if

$$\ell \leq x \leq u \quad \text{and} \quad \forall i \in \{1, \ldots, \mu\} : \gamma_i^t x \equiv 0 \mod T.$$

Decomposing $\gamma_i = \gamma_{i,+} - \gamma_{i,-}$ into positive resp. negative part $\gamma_{i,+}, \gamma_{i,-} \in \{0, 1\}^{A(G)}$, the modulo constraints are equivalent to

$$\forall \in \{1, \ldots, \mu\} : \quad \gamma_i^t x = T z_i, \quad \left\lceil \frac{\gamma_{i,+}^t \ell - \gamma_{i,-}^t u}{T} \right\rceil \leq z_i \leq \left\lfloor \frac{\gamma_{i,+}^t u - \gamma_{i,-}^t \ell}{T} \right\rfloor, \quad z_i \in \mathbb{Z}.$$

These are the so-called *cycle inequalities* (M. A. Odijk, 1994). In particular, for each i, one has to check at most

$$\left\lfloor \frac{\gamma_{i,+}^t u - \gamma_{i,-}^t \ell}{T} \right\rfloor - \left\lceil \frac{\gamma_{i,+}^t \ell - \gamma_{i,-}^t u}{T} \right\rceil + 1 \leq \frac{(\gamma_{i,+} + \gamma_{i,-})^t (u - \ell)}{T} + 1$$

values for z_i. Since, as in the proof of Theorem 7.2.6, we can assume w.l.o.g. that $u - \ell < T$, we have the estimate

$$z_i \leq |\{a \in A(G) : \gamma_{i,a} \neq 0\}| + 1 \leq n + 1,$$

as the γ_i are simple cycles and hence contain at most n vertices.

The description of the polynomial-time algorithm is now: Enumerate all $O((n+1)^\mu)$ integral vectors $(z_1, \ldots, z_\mu)$ satisfying the cycle inequalities and solve the problem

$$\text{Minimize} \quad w^t x \quad \text{subject to} \quad \ell \leq x \leq u \ \text{ and } \ \forall \in \{1, \ldots, \mu\} : \gamma_i^t x = T z_i.$$

This is a minimum cost network tension problem and can be solved in polynomial time by network flow approaches (Hadjiat and Maurras, 1997; Nachtigall and Opitz, 2008). Alternatively, the above minimization problem can be solved by linear programming. □

It remains open whether T-PESP-Optimality can be solved with a fixed-parameter algorithm w.r.t. the cyclomatic number. The main problem is that the cyclomatic number does not bound the number of vertices. However, if, e.g., one additionally fixes the *diameter* of a graph, i.e., the maximum length of an undirected shortest path between two vertices, then also T-PESP-Optimality becomes fixed-parameter tractable:

Corollary 7.5.7. *T-PESP-Optimality is fixed-parameter tractable when parameterized by cyclomatic number and diameter.*

Proof. We adapt the proof of Theorem 7.5.6. In our final estimate of z_i, the number of activities contained in γ_i can be bounded from above by $2d$ if d denotes the diameter of the graph. □

We want to remark that fixing both cyclomatic number and diameter does *not* bound the number of vertices:

Lemma 7.5.8. *For any $k \in \mathbb{N}$, there is an infinite class of simple connected graphs of diameter at most 2 and cyclomatic number at most k.*

Proof. For $r \geq k$, let G be a star graph on r leaves. Connect k distinct pairs of leaves by an edge. Then $\mu(G) = (k+r) - (r+1) + 1 = k$ and G has diameter 2. □

7.5.2. Vertex Cover Number

Definition 7.5.9. For a graph G, its *vertex cover number* $\mathrm{vc}(G)$ is defined as the minimum cardinality of a vertex cover of G.

Lemma 7.5.10 (Fiala et al., 2011, §2)**.** *Let G be a graph. Then* $\mathrm{tw}(G) \leq (G) + 1$.

The vertex cover number does neither limit the number of vertices (consider star graphs) nor the cyclomatic number (complete bipartite graphs $K_{2,q}$), but it does bound the diameter: On a simple path of length d, at least $d/2$ vertices have to be part of a vertex cover. It follows that fixing both cyclomatic number and vertex cover number gives a fixed-parameter algorithm for T-PESP-Optimality by Corollary 7.5.7. We show in the following that T-PESP-Feasibility is $W[1]$-hard when parameterized by the vertex cover number only, so that the existence of a fixed-parameter algorithm is unlikely. It remains unclear if T-PESP-Optimality admits a polynomial-time algorithm for fixed vertex cover number.

Theorem 7.5.11. *T-PESP-Feasibility is $W[1]$-hard when parameterized by the vertex cover number.*

Proof. We provide a fixed-parameter reduction of the List Coloring problem, which is known to be $W[1]$-hard (Fiala et al., 2011, Theorem 1). An instance consists of a graph H together with a finite list $L(v) \subseteq \mathbb{Z}_{\geq 0}$ for each $v \in V(H)$, and the task is to find a vertex coloring of H such that each vertex v is colored with a color in $L(v)$.

Given (H, L), we construct a T-PESP-Feasibility instance (G, T, ℓ, u) as follows: Define $T := \max_{v \in V(H)} L(v) + 1$. Let G be any orientation of H. For each edge of G, the corresponding activity a in $A(G)$ obtains the bounds $\ell_a := 1$ and $u_a := T - 1$. Further add a new vertex v_0 to G. Let $v \in V(H)$ with $L(v) = \{c_1, \dots, c_r\}$, $c_1 < \cdots < c_r$. Add r parallel activities $a_1, \dots, a_r$ from v_0 to v, with bounds

$$\ell_{a_i} := c_i, \quad u_{a_i} := c_{i-1} + T, \quad i = 1, \dots, r,$$

where we set $c_0 := c_r - T$. This way, we model the disjunctive constraints of choosing a color in $L(v)$ as a PESP instance (Christian Liebchen and Möhring, 2007, §3.3).

We claim that (H, L) has a feasible list coloring if and only if (G, T, ℓ, u) admits a feasible periodic timetable. Thus, let $\pi \in [0, T)^{V(H)}$ be a list coloring for (H, L). If $ij \in A(H)$, then $\pi_j \neq \pi_j$, so that $[\pi_j - \pi_i - 1]_T \leq T - 2$ is a feasible periodic slack. Extend π to a timetable on G by setting $\pi_{v_0} := 0$. Let $v \in V(H)$, and assume that v is

colored with the j-th color from its list, i.e., $\pi_v = c_j \in L(v)$. For $i \in \{1, \dots, r\}$, for the i-th activity a_i from v_0 to v, the periodic tension would be

$$[\pi_v - \pi_{v_0} - \ell_{a_i}]_T + \ell_{a_i} = [c_j - c_i]_T + c_i = \begin{cases} c_j & \text{if } i \leq j, \\ c_j + T & \text{if } i > j. \end{cases}$$

In the first case $c_j \leq T \leq c_{i-1} + T$, and in the second $c_i > c_j$ implies $c_j + T \leq c_{i-1} + T$. We conclude that π is a feasible periodic timetable.

Conversely, consider a feasible periodic timetable $\pi \in [0, T)^{V(G)}$. By a shift replacing π by $[\pi - \pi_{v_0}]_T$, we can assume that $\pi_{v_0} = 0$. By restriction, using that $\pi_j \neq \pi_i$ for all $ij \in A(H)$, π yields a coloring of H. It remains to check that this is a feasible list coloring. The periodic tension on an activity a_i from v_0 to v must satisfy

$$[\pi_v - \pi_{v_0} - \ell_{a_i}]_T + \ell_{a_i} = [\pi_v - c_i]_T + c_i \leq c_{i-1} + T \quad \text{for all } i \in \{1, \dots, r\}.$$

Suppose that there is an index j such that $c_j \leq \pi_v < c_{j+1}$. Then the above inequality for $i = j + 1$ means

$$\pi_v + T = [\pi_v - c_{j+1}]_T + c_{j+1} \leq c_j + T,$$

hence $\pi_v = c_j$. If $\pi_v \geq c_r$, then $\pi_v \leq c_0 + T = c_r$. Finally if $\pi_v < c_1$, then $\pi_v \leq c_r - T < 0$, contradicting $\pi_v \geq 0$. This shows that $\pi_v \in \{c_1, \dots, c_r\}$, so that π is indeed a feasible list coloring.

Since all arcs in G not present in H are connected to v_0, we obtain $(G) \leq (H) + 1$. In particular, any fixed-parameter algorithm for T-PESP-FEASIBILITY yields a fixed-parameter algorithm for LIST COLORING. □

7.6. Structure of Realistic Event-Activity Networks

In this section, we discuss the size of the so far discussed graph parameters on realistic periodic timetabling instances. We consider networks with a special structure based on line networks. This structure is the direct outcome of a typical modeling process (Nachtigall, 1998; Christian Liebchen and Möhring, 2007; Schöbel, 2017; Pätzold et al., 2017). For example, the railway networks in the benchmarking library *PESPlib* (Goerigk, 2012) are found as subgraphs of networks with this structure. For this type of networks, we give lower and upper bounds on the branchwidth in terms of the underlying line network. We use this theoretical result to compute bounds on the branchwidth of the smallest PESPlib instance R1L1.

7.6.1. Line-Based Event-Activity Networks

Public transportation systems of cities, but also railway services, are typically organized in *lines*.

Definition 7.6.1. A *line network* $(N, \mathcal{L})$ is a directed multigraph N, together with a set $\mathcal{L}$ of directed walks on G such that the arc set $A(N)$ is the disjoint union of $A(\ell)$ over all *lines* $\ell \in \mathcal{L}$.

Line networks reflect the maps public transport companies offer for passenger information, displaying stations and lines. Depending on the precise application, lines may also constitute non-simple paths or contain cycles (e.g., London's Circle Line or Berlin's Ringbahn). In the context of line planning, we interpret a line network as a frequency-expanded line plan, i.e., some lines might have the same vertex sequence. Given a line network $(N, \mathcal{L})$, construct an event-activity network G as follows:

1. For each line $\ell \in \mathcal{L}$ and each arc $ij \in A(\ell)$, create a *departure event* (i, ℓ, dep) and an *arrival event* (j, ℓ, arr), and connect these by a *driving activity* $((i, \ell, \text{dep}), (j, \ell, \text{arr}))$.

2. For each vertex $i \in V(N)$ and each line $\ell \in \mathcal{L}$, add a *dwelling activity* $((i, \ell, \text{arr}), (i, \ell, \text{dep}))$ if both events exist.

3. For each vertex $i \in V(N)$ and each pair (ℓ_1, ℓ_2) of distinct lines, add a *transfer activity* $((i, \ell_1, \text{arr}), (i, \ell_2, \text{dep}))$ if both events exist.

Definition 7.6.2. An event-activity network G is *line-based* if it arises from a line network $(N, \mathcal{L})$ by the above construction. Shortly, G is *based* on $(N, \mathcal{L})$.

Denote by $\deg^+(i)$ resp. $\deg^-(i)$ the number of outgoing resp. ingoing arcs at i. We summarize some straightforward structural properties of line-based networks in the following lemma:

Lemma 7.6.3. *Let G be based on $(N, \mathcal{L})$.*

1. *G is bipartite, the parts being the departure and arrival events, respectively.*

2. *Every departure event has a unique outgoing activity and every arrival event has a unique ingoing activity. In both cases, these are driving activities.*

3. *The driving activities in G form a perfect matching in G.*

4. *Deleting the driving activities from G and undirecting the arcs results in the disjoint union of complete bipartite graphs $K_{\deg^+(i),\deg^-(i)}$ over $i \in V(N)$.*

7.6.2. Branchwidth of Line-Based Networks

To give bounds on the branchwidth of line-based event-activity networks, we start with a well-known result on the branchwidth of minors:

Theorem 7.6.4 (Neil Robertson and P.D Seymour, 1991, 4.1)**.** *If G is a graph and H is a minor of G, then* $\text{bw}(H) \leq \text{bw}(G)$.

By Lemma 7.6.3, this implies that if G is based on $(N, \mathcal{L})$, then

$$\text{bw}(G) \geq \max_{i \in V(N)} \text{bw}(K_{\deg^+(i),\deg^-(i)}).$$

As we did not manage to find a reference in the literature for the branchwidth of complete bipartite graphs, we give a proof here:

Lemma 7.6.5. *The complete bipartite graph $K_{p,q}$ has branchwidth* $\min(p, q)$.

Proof. Assume $p \leq q$. Let P and Q denote the two parts, $|P| = p$, $|Q| = q$. The vertex separator associated to any neighborhood $\delta(w)$ for $w \in Q$ is given by P. Moreover, if $E \subsetneq \delta(w)$ is a proper subset, then the cardinality of the corresponding vertex separator is $|E| + 1 \leq p$. Take any ternary tree with q leaves labeled with the vertices in Q. Then replace each leaf w by any ternary tree with p leaves, labeled by the edges in $\delta(w)$. The result is a branch decomposition of $K_{p,q}$ of width p. This shows $\mathrm{bw}(K_{p,q}) \leq p$.

To show that $\mathrm{bw}(K_{p,q}) \geq p$, we make use of *tangles*. That is, if we can find a collection $\mathcal{T}$ of subsets of $E(K_{p,q})$ such that

1. for each $A \in \mathcal{T}$, the size of the corresponding vertex separator is at most $p - 1$,
2. for each $A \subseteq E(K_{p,q})$ inducing a separator of size $\leq p - 1$, either A or its complement is in $\mathcal{T}$,
3. for any three sets $A_1, A_2, A_3 \in \mathcal{T}$, their union is not $E(K_{p,q})$,
4. for each $A \in \mathcal{T}$, the subgraph induced by A does not contain all vertices of $K_{p,q}$,

then $\mathrm{bw}(K_{p,q}) \geq p$ or $p \leq 2$ holds (Neil Robertson and P.D Seymour, 1991, p. 4.3). Clearly, $\mathrm{bw}(K_{1,q}) = 1$, as these are star graphs, and $\mathrm{bw}(K_{2,q}) = 2$, as these are series-parallel and contain cycles. Hence suppose $p \geq 3$ and define

$$\mathcal{T} := \{A \subseteq E(K_{p,q}) \mid \mathrm{sep}(A) \leq p-1, |V(K_{p,q}[A]) \cap P| \leq p-1, |V(K_{p,q}[A]) \cap Q| \leq p-1\},$$

where $\mathrm{sep}(A)$ denotes the cardinality of the vertex separator associated to A, and $K_{p,q}[A]$ is the subgraph induced of $K_{p,q}$ by A. We check the above properties:

1. This is clear.
2. Let $A \subseteq E(K_{p,q})$ with $\mathrm{sep}(A) \leq p - 1$ and suppose that $K_{p,q}[A]$ contains all vertices of P. Then there must be a vertex $v \in P$ incident to some edge of A, but not contained in the separator. Hence, A must contain $\delta(v)$, and every vertex $w \in Q$ is incident to the edge $vw \in A$. Similarly, if A contains at least p vertices from Q, then it contains $\delta(w)$ for at least $|V(K_{p,q}[A]) \cap Q| - (p - 1)$ vertices $w \in Q$.
   This means if $\mathrm{sep}(A) \leq p - 1$ and $A \notin \mathcal{T}$, then A contains $\delta(v)$ for some $v \in P$ and $\delta(w)$ for at least $q - (p - 1)$ vertices $w \in Q$. The subgraph induced by the complement of A hence does not contain v, and it also does not contain at least $q - (p - 1)$ vertices of Q. Therefore the complement of A is in $\mathcal{T}$.
3. It follows by the argument in 2. that for each $A \in \mathcal{T}$, the vertex separator is given by $V(K_{p,q}[A])$. Now $K_{p,q}[A]$ is a simple bipartite graph on at most $p - 1$ vertices. It follows that $|A| \leq (p - 1)^2/4$. Now if $A_1, A_2, A_3 \in \mathcal{T}$, the cardinality of their union is at most $3(p - 1)^2/4 < p^2 \leq pq = |E(K_{p,q})|$.
4. This follows as $|V(K_{p,q}[A])| \leq 2p - 2 < 2p \leq p + q = |V(K_{p,q})|$.

Hence we conclude $\mathrm{bw}(K_{p,q}) = p$. □

Theorem 7.6.6. *Let G be a line-based event activity network, based on the line network $(N, \mathcal{L})$. Then*

$$\max_{i \in V(N)} \min(\deg^+(i), \deg^-(i)) \leq \mathrm{bw}(G) \leq \mathrm{cw}(N),$$

and both bounds are sharp.

Recall from Subsection 7.3.4 that $\mathrm{cw}(N)$ denotes the carvingwidth of N, and that

$$\mathrm{cw}(N) \geq \max_{i \in V(N)} (\deg^+(i) + \deg^-(i)).$$

Speaking more intuitively, the carvingwidth is hence bounded by the maximum number of lines departing and arriving at a stop of the line network. In practice, line networks are often planar, and the carvingwidth of planar graphs can be computed in polynomial time by the ratcatcher algorithm (P. D. Seymour and Thomas, 1994).

Proof. The lower bound follows from Lemma 7.6.3, Theorem 7.6.4 and Lemma 7.6.5. For $r \in \mathbb{N}$, let N_r be a graph on the vertex set $\{0, 1, \ldots, 2r\}$, and arcs $(i, 0)$ for $i \in \{1, \ldots, r\}$ and $(0, i)$ for $i \in \{r+1, \ldots, 2r\}$. The lines are given by $\mathcal{L} := \{(i, 0, i+r) \mid i \in \{1, \ldots, r\}\}$. The resulting line-based event-activity network G_r has the structure of a complete bipartite graph $K_{r,r}$ plus $2r$ activities connecting the $K_{r,r}$ with events of degree 1 each. It follows that

$$\mathrm{bw}(G_r) = r = \max_{i \in V(N_r)} \min(\deg^+(i), \deg^-(i)).$$

As N_r is a star graph on $2r$ rays, its carvingwidth is easily determined to be $2r$. Hence, we found a family of graphs for which the lower bound is sharp, and the upper bound is larger than the lower bound.

Concerning the upper bound, we first partition the activities of G: For each $i \in V(N)$, let A_i denote the set of all activities incident to some departure event at i. Then $\{A_i \mid i \in V(N)\}$ partitions $A(G)$ because of the bipartite structure of G. For $i \in V(N)$, let $(\mathcal{B}_i, \varphi_i)$ be a branch decomposition of the subgraph of G induced by A_i. Add a root b_i to each $\mathcal{B}_i$ by subdividing an arbitrary edge. Let $(\mathcal{C}, \psi)$ be an optimal carving decomposition of N. Attach to every leaf v of $\mathcal{C}$ the tree $B_{\psi(v)}$, identifying v with $b_{\psi(v)}$. This results in a branch decomposition $(\mathcal{B}, \varphi)$ of G.

Now let $e \in E(\mathcal{B})$ and let $A(G) = A_e^1 \dot\cup A_e^2$ be the induced partition. If $e \in E(\mathcal{B}_i)$, then one of A_e^1, A_e^2 is contained in A_i, so that the vertex separator S_e has size at most $\deg^+(i) + \deg^-(i)$: S_e contains at most all $\deg^-(i)$ arrival events at i, and every other vertex in S_e must be either a departure event or the unique arrival event following a departure event. Observe that $\deg^+(i) + \deg^-(i) \leq \mathrm{cw}(N)$ due to Theorem 7.3.16. In the other case that $e \in E(\mathcal{C})$, there is a subset $W_e \subseteq V(G)$ such that $A_e^1 = \bigcup_{i \in W_e} A_i$. Each vertex of the vertex separator S_e is hence an arrival event at some $i \in W_e$. The set S_e is in bijection to $\delta(W_e)$ by mapping (j, ℓ, arr) to $ij \in A(\ell) \subseteq A(N)$, where (i, ℓ, dep) is the unique driving activity entering (j, ℓ, arr). Hence, as $\mathcal{C}$ was chosen to be optimal, $|S_e| = |\delta(W_e)| \leq cw(N)$.

Parameter	Value
no. of vertices	3664
no. of arcs	6385
cyclomatic number	2722
vertex cover number	1832
diameter	88
maximum degree	26
branchwidth	$\in [58, 70]$
treewidth	$\in [57, 97]$
carvingwidth	$\in [29, 1820]$

Table 7.1.: Parameters of R1L1

Finally, we show that the upper bound is sharp: For $r \in \mathbb{N}$, let N'_r be a directed simple cycle on r vertices. Then $\mathrm{cw}(N'_r) = 2$. The event-activity network G'_r based on N'_r is then a directed simple cycle on $2r$ vertices and has branchwidth 2. On the other hand, the lower bound does not equal 2, as $\max_{i \in V(N'_r)} \min(\deg^+(i), \deg^-(i)) = 1$. □

7.6.3. Parameters of R1L1

The benchmarking library PESPlib contains 20 difficult T-PESP-OPTIMALITY instances, created with the LinTim toolbox with data of the German railway network (Goerigk, Schachtebeck, et al., 2013). None of the instances is currently solved to proven optimality.

For the event-activity network G of the smallest PESPlib instance R1L1, the values of the parameters discussed in this paper are summarized in Table 7.1. Most of the parameters are easy to obtain, therefore we elaborate only on the width parameters.

An upper bound on branchwidth

The network G in its original shape does not satisfy the properties of Lemma 7.6.3. However, with small modifications, we can find a line network N such that G is a subgraph of an event-activity network based on N.

Transfer activities $a \in A(G)$ are in practice typically recognized by a large span $u_a - \ell_a$. As the period time is $T = 60$, we let $A_t := \{ij \in A(G) \mid u_a - \ell_a \geq 59\}$. We call any vertex i with $ij \in A_t$ for some j an arrival event, and analogously any vertex j with $ij \in A_t$ for some i is called a departure event. This is well-defined and extends to a bipartition of G into arrival resp. departure events.

Now we interpret any activity from a departure to an arrival is called a driving activity. The set of driving activities is not quite a perfect matching in G: There are two arrival events (84 and 256) with two ingoing driving activities, and two arrival events (53 and 177) with no ingoing driving activity. This is due to four mysterious activities with lower and upper bound 0 breaking the structure at this particular spot, see

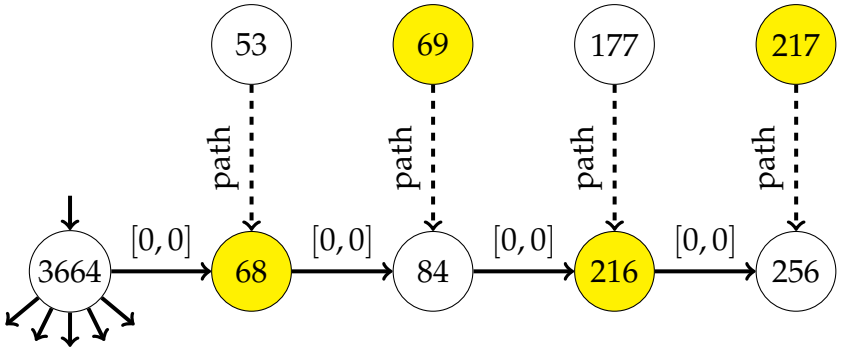


Figure 7.6.: Removed part of R1L1, events recognized as departures are marked yellow

Figure 7.6. However, the vertices 53 to 84 and 177 to 256 can be removed from G by sequentially deleting vertices of degree 1. Since the remaining network has branchwidth at least two, removing vertices of degree 1 has no effect on branchwidth:

Lemma 7.6.7. *Let G be a connected graph, and let $v \in V(G)$ be a vertex of degree 1. If $\mathrm{bw}(G \setminus \{v\}) \geq 2$, then $\mathrm{bw}(G) = \mathrm{bw}(G \setminus \{v\})$.*

Proof. By Theorem 7.6.4, $\mathrm{bw}(G \setminus \{v\}) \leq \mathrm{bw}(G)$. For the reverse inequality consider a branch decomposition $(\mathcal{B}, \varphi)$ of $G \setminus \{v\}$. Since G is connected, v is adjacent to some vertex w of degree ≥ 2. Choose an edge $e \neq \{v, w\}$ incident with w, and replace the leaf of $\mathcal{B}$ corresponding to e by a node with the two children e and $\{v, w\}$. This is a branch decomposition of G. The size of the vertex separator corresponding to $\{v, w\}$ is 1, the size of the separator w.r.t. e is at most 2, and all other vertex separators remain unchanged. This shows $\mathrm{bw}(G) \leq \mathrm{bw}(G \setminus \{v\})$. □

With this adjustment, G has 3552 vertices and 6273 arcs. G satisfies now properties 1-3 of Lemma 7.6.3. Deleting the driving activities yields a disjoint union of bipartite graphs G_i, but they are not all complete. We define N now as the network obtained from G by contracting all these bipartite graphs G_i to a single vertex i. Then G is a subgraph of an event-activity network based on N, choosing, e.g., $\mathcal{L}$ as the set of all single-arc walks. By Theorem 7.6.4 and Theorem 7.6.6, $\mathrm{bw}(G) \leq \mathrm{cw}(N)$.

The network N obtained in this way is unfortunately not planar. The maximum degree in N is 62, so that $\mathrm{cw}(N) \geq 62$. To compute an upper bound, we first preprocess N by removing vertices of degree 2, as this does not alter carvingwidth (Belmonte et al., 2013, Lemma 6). We use then the Kuratowski subgraph detection algorithm implemented in the Python package *networkx* (Hagberg et al., 2008) to recursively remove (multi-)arcs from N until the graph becomes planar. We prefer arcs of low multiplicity, and a sequence of 20 removals of simple arcs finally yields a planar graph N'. We implemented the ratcatcher method of Robertson and Seymour to compute $\mathrm{cw}(N') = 62$ and an optimal carving decomposition. As $V(N') = V(N)$, this is also a carving decomposition of N, but the width increases to 70. We hence conclude $\mathrm{cw}(N) \in [62, 70]$ and $\mathrm{bw}(G) \leq 70$.

A lower bound on branchwidth

Since G is only realized as a subgraph of an event-activity network based on N, we cannot use Theorem 7.6.6 directly. Of course, it remains true that $\mathrm{bw}(G)$ is at least the branchwidth of the (disconnected) subgraph G' obtained by deleting the driving activities. G' is reasonably small, but there seems to be no freely available software for exact branchwidth computations. However, there are treewidth codes, and we use the algorithm by Tamaki (2019), which has been implemented for the PACE 2017 challenge on exact treewidth computations (Dell et al., 2018). It turns out that $\mathrm{tw}(G') = 20$, hence $\mathrm{bw}(G) \geq 14$.

The largest of the components of G' is a bipartite graph with maximum part size 31. If this component were complete, then Theorem 7.6.6 would have predicted $\mathrm{bw}(G) \geq 31$.

To obtain a better bound on $\mathrm{bw}(G)$, we use balanced vertex separators:

Lemma 7.6.8 (N. Robertson and P.D. Seymour, 1995, 3.1)**.** *Let G be a graph. Then there is a vertex separator S with $|S| \leq \mathrm{bw}(G)$ and*

$$\max(|V^1|, |V^2|) \leq \frac{2}{3}|V(G)| - \frac{1}{2}|S|,$$

where $V(G) = V^1 \dot\cup V^2 \dot\cup S$, and no vertex in V^1 is adjacent to a vertex in V^2 and vice versa.

We now compute a minimum cardinality vertex separator subject to the balance constraint of Lemma 7.6.8 by plugging in a straightforward integer program into the CPLEX[1] 12.10 solver. We do not use the full network G as input, but take a smaller network that is obtained after standard preprocessing for T-PESP-Optimality instances (Borndörfer et al., 2019, §3.2). This network is a minor of G, so that we obtain a valid bound on the branchwidth. CPLEX finds a vertex separator of cardinality 58 and is able to solve the instance to optimality. We conclude $\mathrm{bw}(G) \geq 58$.

Treewidth and Carvingwidth

Since $\mathrm{bw}(G) \in [58, 70]$, we obtain by Theorem 7.3.12 that $\mathrm{tw}(G) \in [58, 104]$. As the maximum degree in R1L1 is 26, $\mathrm{cw}(G) \in [29, 1820]$ by Theorem 7.3.16. Determining the exact treewidth of G turns out to be computationally infeasible. We instead use *TCS-Meiji* (Tamaki, 2019) and *FlowCutter* (Hamann and Strasser, 2018), the best two submissions of the PACE 2017 challenge on heuristic treewidth computations (Dell et al., 2018), with different random seeds to obtain a better upper bound on $\mathrm{tw}(G)$. The best bound we could find was $\mathrm{tw}(G) \leq 97$.

Practical Implications

The instance R1L1 has a period time of $T = 60$. It becomes clear from Table 7.1 that neither the dynamic programs of Section 7.4 nor the algorithms presented in Section 7.5

[1] https://www.ibm.com/analytics/cplex-optimizer

can be applied for solving R1L1 in practice. E.g., storing $T^{\mathrm{bw}(G)-1} \geq 60^{57}$ table entries for the branch-decomposition based algorithm 7.4.3 as 32-bit integers would require roughly $9 \cdot 10^{101}$ bytes of space.

7.7. Conclusion

The results of this paper underline that PESP is a notoriously hard problem. Although there are several primal heuristics available, the promising global approaches fail to compute provably optimal solutions: E.g., Mixed-integer programming formulations suffer from weak linear programming relaxations and transformations to Boolean satisfiability problems scale badly. It fits into this picture that exploiting structural parameters such as treewidth does not lead to a polynomial-time algorithm unless P $\neq$ NP. Moreover, the pseudo-polynomial dynamic programs of Section 7.4 are only of theoretical interest. It is even unclear for tentatively large parameters as cyclomatic number and vertex cover number if T-PESP-OPTIMALITY becomes fixed-parameter tractable.

On the positive side, it has been demonstrated in (Lindner and Christian Liebchen, 2019) that balanced edge separators lead to benefits when computing lower bounds of T-PESP-OPTIMALITY instances. We think that this should also carry over to vertex separators, and that good heuristic tree or branch decompositions may be useful as a source for separators in order to tackle PESP by a divide-and-conquer approach.

References

Arnborg, Stefan and Andrzej Proskurowski (1989). "Linear time algorithms for NP-hard problems restricted to partial k-trees". In: *Discrete Applied Mathematics* 23.1, pp. 11–24. ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(89)90031-0.

Belmonte, Rémy et al. (2013). "Characterizing graphs of small carving-width". In: *Discrete Applied Mathematics* 161.13, pp. 1888–1893. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2013.02.036. URL: http://www.sciencedirect.com/science/article/pii/S0166218X13001236.

Bodlaender, Hans L. (1996). "A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth". In: *SIAM Journal on Computing* 25.6, pp. 1305–1317. DOI: 10.1137/S0097539793251219. URL: https://doi.org/10.1137/S0097539793251219.

Bodlaender, Hans L. and Babette van Antwerpen-de Fluiter (2001). "Parallel Algorithms for Series Parallel Graphs and Graphs with Treewidth Two". In: *Algorithmica* 29, pp. 534–559.

Bodlaender, Hans L. and Dimitrios M. Thilikos (1997). "Constructive linear time algorithms for branchwidth". In: *Automata, Languages and Programming*. Ed. by Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 627–637. ISBN: 978-3-540-69194-5.

Borndörfer, Ralf, Niels Lindner, and Sarah Roth (2019). "A concurrent approach to the Periodic Event Scheduling Problem". In: *Journal of Rail Transport Planning & Management*, pp. 100–175. ISSN: 2210-9706. DOI: https://doi.org/10.1016/j.jrtpm.2019.100175. URL: http://www.sciencedirect.com/science/article/pii/S2210970619300769.

Dell, Holger et al. (2018). "The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration". In: *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*. Ed. by Daniel Lokshtanov and Naomi Nishimura. Vol. 89. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 30:1–30:12. ISBN: 978-3-95977-051-4. DOI: 10.4230/LIPIcs.IPEC.2017.30. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8558.

Eppstein, David (2018). "The Effect of Planarization on Width". In: *Journal of Graph Algorithms and Applications* 22.3, pp. 461–481. DOI: 10.7155/jgaa.00468.

Fiala, Jiří, Petr A. Golovach, and Jan Kratochvíl (2011). "Parameterized complexity of coloring problems: Treewidth versus vertex cover". In: *Theoretical Computer Science* 412.23. Theory and Applications of Models of Computation (TAMC 2009), pp. 2513–2523. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.10.043.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability*. W. H. Freeman and Company, San Francisco.

Goerigk, Marc (2012). *PESPlib – A benchmark library for periodic event scheduling*. http://num.math.uni-goettingen.de/~m.goerigk/pesplib/.

Goerigk, Marc, Michael Schachtebeck, and Anita Schöbel (2013). "Evaluating line concepts using travel times and robustness". In: *Public Transport* 5, pp. 267–284.

Hadjiat, Malika and Jean François Maurras (1997). "A strongly polynomial algorithm for the minimum cost tension problem". In: *Discrete Mathematics* 165-166. Graphs and Combinatorics, pp. 377–394. ISSN: 0012-365X. DOI: https://doi.org/10.1016/S0012-365X(96)00185-9. URL: http://www.sciencedirect.com/science/article/pii/S0012365X96001859.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart (2008). "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, pp. 11–15.

Hamann, Michael and Ben Strasser (Feb. 2018). "Graph Bisection with Pareto Optimization". In: *J. Exp. Algorithmics* 23. ISSN: 1084-6654. DOI: 10.1145/3173045. URL: https://doi.org/10.1145/3173045.

Heuven van Staereling, Irving van (2018). "Tree Decomposition Methods for the Periodic Event Scheduling Problem". In: *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*. Ed. by Ralf Borndörfer and Sabine Storandt. Vol. 65. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 6:1–6:13. ISBN: 978-3-95977-096-5. DOI: 10.4230/OASIcs.ATMOS.2018.6. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9711.

Karp, Richard M. (1972). "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Boston, MA: Springer US, pp. 85–103. ISBN: 978-1-4684-2001-2.

Kloks, Ton (1994). *Treewidth, Computations and Approximations*. Vol. 842. Lecture Notes in Computer Science. Springer. ISBN: 3-540-58356-4. DOI: 10.1007/BFb0045375. URL: https://doi.org/10.1007/BFb0045375.

Liebchen, C. (2006). "Periodic timetable optimization in public transport". PhD thesis. Technische Universität Berlin.

Liebchen, Christian and Rolf H Möhring (2007). "The modeling power of the Periodic Event Scheduling Problem: railway timetables – and beyond". In: *Algorithmic methods for railway optimization*. Springer, pp. 3–40.

Lindner, Niels and Christian Liebchen (2019). "New Perspectives on PESP: T-Partitions and Separators". In: *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*. Ed. by Valentina Cacchiani and Alberto Marchetti-Spaccamela. Vol. 75. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2:1–2:18. ISBN: 978-3-95977-128-3. DOI: 10.4230/OASIcs.ATMOS.2019.2. URL: http://drops.dagstuhl.de/opus/volltexte/2019/11414.

Nachtigall, Karl (1993). *Exact Solution Methods for Periodic Programs*. Tech. rep. 14/93. Hildesheimer Informatikberichte.

— (1998). "Periodic Network Optimization and Fixed Interval Timetables". Habilitation thesis. Universität Hildesheim.

Nachtigall, Karl and Jens Opitz (2008). "Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations". In: *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*. Ed. by Matteo Fischetti and Peter Widmayer. Vol. 9. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz- Zentrum für Informatik. ISBN: 978-3-939897-07-1.

Nestoridis, Nestor V. and Dimitrios M. Thilikos (2014). "Square roots of minor closed graph classes". In: *Discrete Applied Mathematics* 168. Fifth Workshop on Graph Classes, Optimization, and Width Parameters, Daejeon, Korea, October 2011, pp. 34–39. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2013.05.026. URL: http://www.sciencedirect.com/science/article/pii/S0166218X13002552.

Odijk, M. A. (1994). *Construction of periodic timetables, Part 1: A cutting plane algorithm*. Tech. rep. 94-61. TU Delft.

Pätzold, Julius et al. (2017). "Look-Ahead Approaches for Integrated Planning in Public Transportation". In: *ATMOS*. Vol. 59. OASICS. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 17:1–17:16.

Robertson, N. and P.D. Seymour (1995). "Graph Minors. XIII. The Disjoint Paths Problem". In: *Journal of Combinatorial Theory, Series B* 63.1, pp. 65–110. ISSN: 0095-8956.

DOI: https://doi.org/10.1006/jctb.1995.1006. URL: http://www.science%20direct.com/science/article/pii/S0095895685710064.

Robertson, Neil and P.D Seymour (1984). "Graph minors. III. Planar tree-width". In: *Journal of Combinatorial Theory, Series B* 36.1, pp. 49–64. ISSN: 0095-8956. DOI: https://doi.org/10.1016/0095-8956(84)90013-3. URL: http://www.sciencedirect.com/science/article/pii/0095895684900133.

— (1991). "Graph minors. X. Obstructions to tree-decomposition". In: *Journal of Combinatorial Theory, Series B* 52.2, pp. 153–190. ISSN: 0095-8956. DOI: https://doi.org/10.1016/0095-8956(91)90061-N. URL: http://www.science%20direct.com/science/article/pii/009589569190061N.

Schöbel, Anita (2017). "An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation". In: *Transportation Research Part C: Emerging Technologies* 74, pp. 348–365. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2016.11.018. URL: http://www.sciencedirect.com/science/article/pii/S0968090X1630242X.

Schrijver, Alexander (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc. ISBN: 0471908541.

Serafini, Paolo and Walter Ukovich (1989b). "A mathematical model for periodic scheduling problems". In: *SIAM Journal on Discrete Mathematics* 2.4, pp. 550–581.

Seymour, P. D. and R. Thomas (June 1994). "Call routing and the ratcatcher". English (US). In: *Combinatorica* 14.2, pp. 217–241. ISSN: 0209-9683. DOI: 10.1007/BF01215352.

Tamaki, Hisao (May 2019). "Positive-Instance Driven Dynamic Programming for Treewidth". In: *J. Comb. Optim.* 37.4, pp. 1283–1311. ISSN: 1382-6905. DOI: 10.1007/s10878-018-0353-z. URL: https://doi.org/10.1007/s10878-018-0353-z.

Part III.

Delay Propagation in Railway Networks

8. Modeling Delay Propagation and Transmission in Railway Networks

Chapter is omitted due to copyright.

9. Discerning Primary and Secondary Delays in Railway Networks using Explainable AI

Chapter is omitted in the online version due to copyright.

10. Black-Box Optimization in Railway Simulations

Chapter is omitted in the online version due to copyright.

11. Conclusion

In this cumulative dissertation, we studied selected aspects in railway timetable optimization. Let us now conclude the core findings of the contributions and name possible directions for further research. We grouped the contributions into the three dimensions of Practical Applications of Automatic Railway Timetabling, Algorithms and Computability in Timetable Generation and Delay Propagation in Railway Networks. In the papers of the first dimension, we developed a column generation approach for scheduling multiple trains simultaneously. In the computational experiments, we were able to schedule more than 5000 freight trains in Germany and thus contributed to the field by extending the applications scope. For the pricing problem in the column generation, we applied a slot construction algorithm that we furthermore used for measuring the railway network capacity. In particular, we computed in a case study that when equipping freight trains with electro-pneumatic brakes and middle buffer couplings, the capacity grows by 4%. This method for measuring railway network capacity is the second research contribution of the first dimension. As a possible outlook of the first contribution, we see the role-out to plan the annual timetable automatically. In terms of compatibility, we claim that the column generation approach is also applicable for multi-day planning when slots differ between different days. The open research question here is to partition the requested operation periods of the trains so that there are no conflicts between any two trains on any day within the year. In the second contribution, we proposed a novel method for measuring railway capacities. Other methods quantify capacity as a function that depends on the amount of follow-up delays that are allowed. Future work for this research branch should therefore incorporate operational quality and count the number of constructed slots such that the expected follow-up delays are bounded.

For the second dimension, we developed local search heuristic algorithms for solving large scale MaxSAT problems. MaxSAT is a problem class that occurs both in multiple train scheduling and periodic scheduling. We could prove that the local search algorithms perform competitive compared to state-of-the-art solvers from the 2019 MaxSAT evaluation. Moreover, we proved that the Periodic Event Scheduling problem is NP-complete even for graphs of bounded treewidth, branchwidth and carving-width. Unless $P = NP$, these statements prove that there is no polynomial algorithm parameterized by any of these structural graph parameters for the PESP. Hence, we conject that solving the PESP should rather be tackled by heuristic approaches. As we made good experiences with local search techniques for solving the Independent Set problem and MaxSAT, developing a local search PESP heuristic could be a possible research direction.

In the third dimension, we developed a framework for deriving delay transmission rules from historical railway operations data. In selected stations, we could identify train interactions and conclude necessary timetable adjustments that prevent the propagation of delays. Furthermore, we proposed to apply the SHAP value method from Explainable AI for the application of discerning primary from secondary causes in the railway delay data. Finally, we presented black-box optimization rules for robust timetable optimization in a simulation framework. This method opens the opportunity for any simulation approach to consider it a black-box and incorporate optimization rules. We see the open task of this dimension to unify these approaches in a proper simulation environment. That is, discern primary from secondary delay distributions in the historical data at first. Then, use the primary delay distributions for simulating the stochastically occurring disturbances in the operations. Finally, adjust the timetable with black-box optimization rules. The optimization rules can be based on the transmission rules that are detected between the secondary delay data and the evaluation is performed by the simulation.

Global References

Amaran, Satyajith et al. (2017). "Simulation optimization: A review of algorithms and applications". In: *CoRR* abs/1706.08591. URL: http://arxiv.org/abs/1706.08591.

Andrade, Diogo Vieira, Mauricio G. C. Resende, and Renato F. Werneck (2012). "Fast local search for the maximum independent set problem". In: *J. Heuristics* 18.4, pp. 525–547. DOI: 10.1007/s10732-012-9196-4. URL: https://doi.org/10.1007/s10732-012-9196-4.

Andrade, Diogo Vieira, Mauricio G. C. Resende, and Renato Fonseca F. Werneck (2012). "Fast local search for the maximum independent set problem". In: *J. Heuristics* 18.4, pp. 525–547.

Arnborg, Stefan and Andrzej Proskurowski (1989). "Linear time algorithms for NP-hard problems restricted to partial *k*-trees". In: *Discrete Applied Mathematics* 23.1, pp. 11–24. ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(89)90031-0.

Audemard, Gilles and Laurent Simon (2009). "Predicting Learnt Clauses Quality in Modern SAT Solvers". In: *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. IJCAI'09. Pasadena, California, USA, pp. 399–404.

Bacchus, Fahiem, Jeremias Berg, et al., eds. (2020). *MaxSAT Evaluation 2020: Solver and Benchmark Descriptions*. English. Vol. B-2020-2. Department of Computer Science Report Series B. Finland: Department of Computer Science, University of Helsinki.

Bacchus, Fahiem, Matti Järvisalo, and Ruben Martins (Sept. 2019). "MaxSAT Evaluation 2018: New Developments and Detailed Results". In: *Journal on Satisfiability, Boolean Modeling and Computation* 11, pp. 99–131. DOI: 10.3233/SAT190119.

Barrett, C., R. Jacob, and M. Marathe (2000). "Formal-Language-Constrained Path Problems". In: *SIAM Journal on Computing* 30.3, pp. 809–837.

Belmonte, Rémy et al. (2013). "Characterizing graphs of small carving-width". In: *Discrete Applied Mathematics* 161.13, pp. 1888–1893. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2013.02.036. URL: http://www.sciencedirect.com/science/article/pii/S0166218X13001236.

Belov, Anton, António Morgado, and Joao Marques-Silva (2013). "SAT-Based Preprocessing for MaxSAT". In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by Ken McMillan, Aart Middeldorp, and Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 96–111. ISBN: 978-3-642-45221-5.

Berg, Jeremias, Emir Demirović, and Peter J. Stuckey (2019). "Core-Boosted Linear Search for Incomplete MaxSAT". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Ed. by Louis-Martin Rousseau and Kostas Stergiou. Cham: Springer International Publishing, pp. 39–56. ISBN: 978-3-030-19212-9.

Berg, Jeremias, Antti Hyttinen, and Matti Jarvisalo (2019). "Applications of MaxSAT in Data Analysis". In: *Proceedings of Pragmatics of SAT 2015 and 2018*. Ed. by Daniel Le

Berre and Matti Jarvisalo. Vol. 59. EPiC Series in Computing. EasyChair, pp. 50–64. DOI: 10.29007/3qkh. URL: https://easychair.org/publications/paper/6HpF.

Berg, Jeremias and Matti Järvisalo (2017). "Cost-optimal constrained correlation clustering via weighted partial Maximum Satisfiability". In: *Artificial Intelligence* 244. Combining Constraint Solving with Mining and Learning, pp. 110–142. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2015.07.001.

Berg, Jeremias, Matti Järvisalo, and Brandon Malone (22–25 Apr 2014). "Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, pp. 86–95.

Bodlaender, Hans L. (1996). "A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth". In: *SIAM Journal on Computing* 25.6, pp. 1305–1317. DOI: 10.1137/S0097539793251219. URL: https://doi.org/10.1137/S0097539793251219.

Bodlaender, Hans L. and Babette van Antwerpen-de Fluiter (2001). "Parallel Algorithms for Series Parallel Graphs and Graphs with Treewidth Two". In: *Algorithmica* 29, pp. 534–559.

Bodlaender, Hans L. and Dimitrios M. Thilikos (1997). "Constructive linear time algorithms for branchwidth". In: *Automata, Languages and Programming*. Ed. by Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 627–637. ISBN: 978-3-540-69194-5.

Bondy, John Adrian, Uppaluri Siva Ramachandra Murty, et al. (1976). *Graph theory with applications*. Macmillan Press.

Bonett, Douglas G and Thomas A Wright (2000). "Sample size requirements for estimating Pearson, Kendall and Spearman correlations". In: *Psychometrika* 65.1, pp. 23–28.

Borndörfer, Ralf, Niels Lindner, and Sarah Roth (2019). "A concurrent approach to the Periodic Event Scheduling Problem". In: *Journal of Rail Transport Planning & Management*, pp. 100–175. ISSN: 2210-9706. DOI: https://doi.org/10.1016/j.jrtpm.2019.100175. URL: http://www.sciencedirect.com/science/article/pii/S2210970619300769.

— (2020). "A Concurrent Approach to the Periodic Event Scheduling Problem". In: *Journal of Rail Transport Planning & Management* 15, pp. 100–175. DOI: 10.1016/j.jrtpm.2019.100175.

Borndörfer, Ralf and Thomas Schlechte (2007). "Models for Railway Track Allocation". In: *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*. Ed. by Christian Liebchen, Ravindra Ahuja, and Juan Mesa. Vol. 07/001.

Bouhmala, Noureddine (Feb. 2019). "Combining simulated annealing with local search heuristic for Max-SAT". In: *Journal of Heuristics* 25.1, pp. 47–69.

Brys, Guy, Mia Hubert, and Anja Struyf (2004). "A robust measure of skewness". In: *Journal of Computational and Graphical Statistics* 13.4, pp. 996–1017.

Büker, Thorsten and Bernhard Seybold (2012). "Stochastic modelling of delay propagation in large networks". In: *Journal of Rail Transport Planning & Management* 2.1-2, pp. 34–50.

Butts, Carter T. (2009). "Revisiting the foundations of network analysis". In: *science* 325.5939, pp. 414–416.

Cacchiani, Valentina, Alberto Caprara, and Paolo Toth (2010). "Non-cyclic train timetabling and comparability graphs". In: *Operations Research Letters* 38.3, pp. 179–184. ISSN: 0167-6377.

Caprara, A. et al. (2002). "Solution of real-world train timetabling problems". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*.

Caprara, Alberto (2015). "Timetabling and assignment problems in railway planning and integer multicommodity flow". In: *Networks* 66.1, pp. 1–10.

Caprara, Alberto, Matteo Fischetti, and Paolo Toth (2002). "Modeling and Solving the Train Timetabling Problem". In: *Operations Research* 50.5, pp. 851–861. ISSN: 0030364X, 15265463.

Carey, Malachy and Andrzej Kwieciński (1994). "Stochastic approximation to the effects of headways on knock-on delays of trains". In: *Transportation Research Part B: Methodological* 28.4, pp. 251–267.

Clarke, Edmund et al. (July 2001). "Bounded Model Checking Using Satisfiability Solving". In: *Formal Methods in System Design* 19.1, pp. 7–34. ISSN: 1572-8102. DOI: 10.1023/A:1011276507260. URL: https://doi.org/10.1023/A:1011276507260.

Cleveland, Robert B, William S Cleveland, and Irma Terpenning (1990). "STL: A seasonal-trend decomposition procedure based on loess". In: *Journal of Official Statistics* 6.1, p. 3.

Cox, David R (1972). "The analysis of multivariate binary data". In: *Applied statistics*, pp. 113–120.

Curchod, Anne (2007). *analyse de la stabilité d'horaires ferroviaires cadencés sur un réseau maillé: Bedienungshandbuch*. Lausanne: FASTA II.

Dahms, Florian et al. (2019). *Transforming automatic scheduling in a working application for a railway infrastructure manager*. Rail Norrköping Conference.

Dantzig, George B. (1963). "Linear Programming and Extensions". In: Princeton University Press, Princeton.

Davis, Martin and Hilary Putnam (July 1960). "A Computing Procedure for Quantification Theory". In: *J. ACM* 7.3, pp. 201–215. ISSN: 0004-5411. DOI: 10.1145/321033.321034.

Dell, Holger et al. (2018). "The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration". In: *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*. Ed. by Daniel Lokshtanov and Naomi Nishimura. Vol. 89. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 30:1–30:12. ISBN: 978-3-95977-051-4. DOI: 10.4230/LIPIcs.IPEC.2017.30. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8558.

Demirović, Emir and Peter J. Stuckey (2019). "Techniques Inspired by Local Search for Incomplete MaxSAT and the Linear Algorithm: Varying Resolution and Solution-

Guided Search". In: *Principles and Practice of Constraint Programming*. Ed. by Thomas Schiex and Simon de Givry. Cham: Springer International Publishing, pp. 177–194. ISBN: 978-3-030-30048-7.

Eppstein, David (2018). "The Effect of Planarization on Width". In: *Journal of Graph Algorithms and Applications* 22.3, pp. 461–481. DOI: 10.7155/jgaa.00468.

Erickson, Jeff (2019). *Lecture Notes on NP-Hardness*. http://jeffe.cs.illinois.edu/teaching/algorithms/book/12-nphard.pdf.

Even, S., A. Itai, and A. Shamir (1975). "On the complexity of time table and multi-commodity flow problems". In: *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pp. 184–193.

Fan, Yi et al. (2016). "Ramp: A Local Search Solver based on Make-positive Variables". In: *MaxSAT Evaluation*.

Fiala, Jiří, Petr A. Golovach, and Jan Kratochvíl (2011). "Parameterized complexity of coloring problems: Treewidth versus vertex cover". In: *Theoretical Computer Science* 412.23. Theory and Applications of Models of Computation (TAMC 2009), pp. 2513–2523. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.10.043.

Fischetti, Matteo and Michele Monaci (2009). "Light Robustness". In: *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Berlin, Heidelberg: Springer-Verlag, pp. 61–84. ISBN: 9783642054648. URL: https://doi.org/10.1007/978-3-642-05465-5_3.

Ford, L. R. and D. R. Fulkerson (1956). "Maximal Flow Through a Network". In: *Canadian Journal of Mathematics* 8, pp. 399–404. DOI: 10.4153/CJM-1956-045-5.

Freeman, Linton C (1978). "Centrality in social networks conceptual clarification". In: *Social networks* 1.3, pp. 215–239.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability*. W. H. Freeman and Company, San Francisco.

Garey, Michael R. and David S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Ghofrani, Faeze et al. (2018). "Recent applications of big data analytics in railway transportation systems: A survey". In: *Transportation Research Part C: Emerging Technologies* 90, pp. 226–246.

Goerigk, Marc (2012). *PESPlib – A benchmark library for periodic event scheduling*. http://num.math.uni-goettingen.de/~m.goerigk/pesplib/.

Goerigk, Marc, Michael Schachtebeck, and Anita Schöbel (2013). "Evaluating line concepts using travel times and robustness". In: *Public Transport* 5, pp. 267–284.

Goerigk, Marc and Anita Schöbel (2010). "An Empirical Analysis of Robustness Concepts for Timetabling". In: *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*. Ed. by Thomas Erlebach and Marco Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 100–113. ISBN: 978-3-939897-20-0. DOI: 10.4230/OASIcs.ATMOS.2010.100. URL: http://drops.dagstuhl.de/opus/volltexte/2010/2753.

Gong, Guanglu, Yong Liu, and Minping Qian (2001). "An adaptive simulated annealing algorithm". In: *Stochastic Processes and their Applications* 94.1, pp. 95–103. ISSN: 0304-4149. URL: https://doi.org/10.1016/S0304-4149(01)00082-5.

Goverde, Rob MP (2010). "A delay propagation algorithm for large-scale railway traffic networks". In: *Transportation Research Part C: Emerging Technologies* 18.3, pp. 269–287.

Großmann, Peter et al. (2012a). "Solving Periodic Event Scheduling Problems with SAT". In: *Advanced Research in Applied Artificial Intelligence*. Ed. by He Jiang et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 166–175. ISBN: 978-3-642-31087-4.

— (2012b). "Solving Periodic Event Scheduling Problems with SAT". In: *Advanced Research in Applied Artificial Intelligence*. Ed. by He Jiang et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 166–175. ISBN: 978-3-642-31087-4.

— (2012c). *Solving Periodic Event Scheduling Problems with SAT*. in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 166-175, Springer.

— (2012d). *Solving Periodic Event Scheduling Problems with SAT*. in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 166-175, Springer.

Großmann, Peter et al. (2016). "Capacity-utilized Integration and Optimization of Rail Freight Train Paths into 24 Hours Timetables". In: *Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transportation Systems*.

Grosso, Andrea, Marco Locatelli, and Wayne Pullan (Dec. 2008). "Simple ingredients leading to very efficient heuristics for the maximum clique problem". In: *Journal of Heuristics* 14.6, pp. 587–612. ISSN: 1572-9397. DOI: 10.1007/s10732-007-9055-x. URL: https://doi.org/10.1007/s10732-007-9055-x.

Guerra, João and Inês Lynce (2012). "Reasoning over Biological Networks Using Maximum Satisfiability". In: *Principles and Practice of Constraint Programming*. Ed. by Michela Milano. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 941–956. ISBN: 978-3-642-33558-7.

Hadjiat, Malika and Jean François Maurras (1997). "A strongly polynomial algorithm for the minimum cost tension problem". In: *Discrete Mathematics* 165-166. Graphs and Combinatorics, pp. 377–394. ISSN: 0012-365X. DOI: https://doi.org/10.1016/S0012-365X(96)00185-9. URL: http://www.sciencedirect.com/science/article/pii/S0012365X96001859.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart (2008). "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, pp. 11–15.

Hamann, Michael and Ben Strasser (Feb. 2018). "Graph Bisection with Pareto Optimization". In: *J. Exp. Algorithmics* 23. ISSN: 1084-6654. DOI: 10.1145/3173045. URL: https://doi.org/10.1145/3173045.

Hansen, Ingo and Jörn Pachl (2014). *Railway Timetabling & Operations. Analysis - Modelling - Optimisation - Simulation - Performance Evaluation.*

Hauck, Florian and Natalia Kliewer (2019). "Big data analytics im Bahnverkehr". In: *HMD Praxis der Wirtschaftsinformatik*. URL: https://doi.org/10.1365/s40702-019-00524-7.

Heuven van Staereling, Irving van (2018). "Tree Decomposition Methods for the Periodic Event Scheduling Problem". In: *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*. Ed. by Ralf Borndörfer and Sabine Storandt. Vol. 65. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 6:1–6:13. ISBN: 978-3-95977-096-5. DOI: 10.4230/OASIcs.ATMOS.2018.6. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9711.

Hoos, Holger H. (1996). "Solving hard combinatorial problems with GSAT — A case study". In: *KI-96: Advances in Artificial Intelligence*. Ed. by Günther Görz and Steffen Hölldobler. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 107–119. ISBN: 978-3-540-70669-4.

Huber, Peter J. and Elvezio M. Ronchetti (Apr. 2009). *Robust statistics*. Second. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc, New York. ISBN: 9780470129906. DOI: 10.1002/9780470434697.

Hubert, Mia and Ellen Vandervieren (2008). "An adjusted boxplot for skewed distributions". In: *Computational statistics & data analysis* 52.12, pp. 5186–5201.

Huisman, Tijs and Richard J. Boucherie (2001). "Running times on railway sections with heterogeneous train traffic". In: *Transportation Research Part B: Methodological* 35.3, pp. 271–292. ISSN: 0191-2615. DOI: https://doi.org/10.1016/S0191-2615(99)00051-X. URL: http://www.sciencedirect.com/science/article/pii/S019126159900051X.

Hyttinen, Antti, Frederick Eberhardt, and Matti Järvisalo (2014). "Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming". In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pp. 340–349.

Jégou, Philippe (1993). "Decomposition of Domains Based on the Micro-structure of Finite Constraint-satisfaction Problems". In: *Proceedings of the Eleventh National Conference on Artificial Intelligence*. AAAI'93. Washington, D.C.: AAAI Press, pp. 731–736. ISBN: 0-262-51071-5. URL: http://dl.acm.org/citation.cfm?id=1867270.1867379.

Karp, Richard M. (1972). "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Boston, MA: Springer US, pp. 85–103. ISBN: 978-1-4684-2001-2.

Kecman, P., F Corman, and L Meng (2015). "Train delay evolution as a stochastic process". English. In: *Proceedings of the 6th International Conference on Railway Operations Modelling and Analysis*. Ed. by N Tomii, CPL Barkan, and al et. IAROR.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (May 1983). "Optimization by Simulated Annealing". In: *Science* 220, pp. 671–680. DOI: 10.1126/science.220.4598.671.

Kloks, Ton (1994). *Treewidth, Computations and Approximations*. Vol. 842. Lecture Notes in Computer Science. Springer. ISBN: 3-540-58356-4. DOI: 10.1007/BFb0045375. URL: https://doi.org/10.1007/BFb0045375.

Kono, Ami, H Yakubi, and Norio Tomii (2016). "Identifying the cause of delays in urban railways using datamining technique". In: *Asian conference on Railway infrastructure and Transportation*, pp. 227–230.

Kümmling, Michael, Peter Großmann, and Jens Opitz (2015). *Maximisation of homogenous rail freight train paths at a given level of quality*. 27th European Conference on Operational Research, Glasgow.

Lei, Zhendong and Shaowei Cai (July 2018). "Solving (Weighted) Partial MaxSAT by Dynamic Local Search for SAT". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 1346–1352. DOI: 10.24963/ijcai.2018/187.

Liebchen, C. (2006). "Periodic timetable optimization in public transport". PhD thesis. Technische Universität Berlin.

Liebchen, Christian (2006). *Periodic Timetable Optimization in Public Transport*. dissertation.de.

— (2008). "The First Optimized Railway Timetable in Practice". In: *Transportation Science* 42.4, pp. 420–435. DOI: 10.1287/trsc.1080.0240. eprint: https://doi.org/10.1287/trsc.1080.0240. URL: https://doi.org/10.1287/trsc.1080.0240.

Liebchen, Christian and Rolf H Möhring (2007). "The modeling power of the Periodic Event Scheduling Problem: railway timetables – and beyond". In: *Algorithmic methods for railway optimization*. Springer, pp. 3–40.

Lindner, Niels and Christian Liebchen (2019). "New Perspectives on PESP: T-Partitions and Separators". In: *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*. Ed. by Valentina Cacchiani and Alberto Marchetti-Spaccamela. Vol. 75. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2:1–2:18. ISBN: 978-3-95977-128-3. DOI: 10.4230/OASIcs.ATMOS.2019.2. URL: http://drops.dagstuhl.de/opus/volltexte/2019/11414.

Lindner, Niels and Julian Reisch (2020). *Parameterized Complexity of Periodic Timetabling*. eng. Tech. rep. 20-15. Takustr. 7, 14195 Berlin: ZIB.

Little, Roderick J. A. and Donald B. Rubin. (2002). *Statistical analysis with missing data*. eng. 2nd ed. Wiley Series in Probability and Statistics. Hoboken, New Jersey: John Wiley & Sons, Inc. ISBN: 1-119-01356-9.

Long, Jeffrey D. and Norman Cliff (1997). "Confidence intervals for Kendall's tau". In: *British Journal of Mathematical and Statistical Psychology* 50.1, pp. 31–41.

Lundberg, Scott M and Su-In Lee (2017). "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems*, pp. 4765–4774.

Maróti, Gábor (July 2017). "A branch-and-bound approach for robust railway timetabling". English. In: *Public Transport* 9.1-2, pp. 73–94. ISSN: 1866-749X. DOI: 10.1007/s12469-016-0143-x.

Martin, Ullrich, Carlo von Molo, et al. (2015). "Umfassende Einführung der Mittelpufferkupplung (in German)". In: *Neues verkehrswissenschaftliches Journal - Ausgabe 13*.

Martin, Ullrich, Niels Neuberg, et al. (2015). "Automatische Mittelpufferkupplung mit elektrischer Leitungsverbindung – Perspektiven für EIU und EVU (in German)". In: *ETR – Eisenbahntechnische Rundschau, Ausgabe 11/2015*.

Molnár, Botond et al. (2018). "A high-performance analog Max-SAT solver and its application to Ramsey numbers". In: *CoRR* abs/1801.06620. arXiv: 1801.06620. URL: http://arxiv.org/abs/1801.06620.

Nachtigall, Karl (1993). *Exact Solution Methods for Periodic Programs*. Tech. rep. 14/93. Hildesheimer Informatikberichte.

— (1998). "Periodic Network Optimization and Fixed Interval Timetables". Habilitation thesis. Universität Hildesheim.

Nachtigall, Karl and Jens Opitz (2008). "Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations". In: *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*. Ed. by Matteo Fischetti and Peter Widmayer. Vol. 9. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz- Zentrum für Informatik. ISBN: 978-3-939897-07-1.

— (2014). *Modelling and Solving a Train Path Assignment Model*. Proceedings of the International Conference on Operations Research, Aachen.

Nestoridis, Nestor V. and Dimitrios M. Thilikos (2014). "Square roots of minor closed graph classes". In: *Discrete Applied Mathematics* 168. Fifth Workshop on Graph Classes, Optimization, and Width Parameters, Daejeon, Korea, October 2011, pp. 34–39. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2013.05.026. URL: http://www.sciencedirect.com/science/article/pii/S0166218X13002552.

Odijk, M. A. (1994). *Construction of periodic timetables, Part 1: A cutting plane algorithm*. Tech. rep. 94-61. TU Delft.

Odijk, Michiel (1994). *Construction of Periodic Timetables: A Cutting Plane Algorithm*. in: Technical Report, TU Delft.

Ohrimenko, Olga, Peter J. Stuckey, and Michael Codish (2007). "Propagation = Lazy Clause Generation". In: *Principles and Practice of Constraint Programming – CP 2007*. Ed. by Christian Bessière. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 544–558. ISBN: 978-3-540-74970-7.

Opsahl, Tore, Filip Agneessens, and John Skvoretz (2010). "Node centrality in weighted networks: Generalizing degree and shortest paths". In: *Social Networks* 32.3, pp. 245–251. ISSN: 0378-8733. DOI: https://doi.org/10.1016/j.socnet.2010.03.006. URL: http://www.sciencedirect.com/science/article/pii/S0378873310000183.

Park, James D. (2002). "Using Weighted Max-SAT Engines to Solve MPE". In: *Proc. 18th Nat'l Conf. Artificial Intelligence*, pp. 682–687.

Pätzold, Julius et al. (2017). "Look-Ahead Approaches for Integrated Planning in Public Transportation". In: *ATMOS*. Vol. 59. OASICS. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 17:1–17:16.

Pisinger, David and Stefan Ropke (2010). "Large Neighborhood Search". In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. Boston, MA: Springer US, pp. 399–419.

Pöhle, Daniel (2016). "Strategische Planung und Optimierung der Kapazität in Eisenbahnnetzen unter Nutzung automatischer Taktfahrplanung (in German)". In: *Disserta Verlag*.

Pöhle, Daniel and Matthias Feil (2016). "Optimierte Belegung von Systemtrassen im industrialisierten Fahrplan: Herausforderungen und erfolgreiche Ansätze (in German)". In: *Tagungsband der 25. Verkehrswissenschaftliche Tage 2016, Dresden*.

Pöhle, Daniel, Anna-Lena Frank, et al. (2020). "Automatisierte Fahrplanerstellung bei der DB Netz (in German)". In: *Tagungsbericht Heureka 2021*.

Reisch, Julian (2020). *State of the Art Overview on Automatic Railway Timetable Generation and Optimization*. eng. URL: http://dx.doi.org/10.17169/refubium-28658.

Reisch, Julian, Peter Großmann, and Natalia Kliewer (2019). "Conflict Resolving - A Maximum Independent Set Heuristics for Solving MaxSAT". In: *Proceedings of the 22nd International Multiconference Information Society* 1, pp. 67–71.

— (2020). "Stable Resolving - A Randomized Local Search Heuristic for MaxSAT". In: *KI 2020: Advances in Artificial Intelligence*. Ed. by Ute Schmid, Franziska Klügl, and Diedrich Wolter. Cham: Springer International Publishing, pp. 163–175. ISBN: 978-3-030-58285-2.

Reisch, Julian, Peter Großmann, Daniel Pöhle, et al. (2021). "Conflict resolving – A local search algorithm for solving large scale conflict graphs in freight railway timetabling". In: *European Journal of Operational Research*. ISSN: 0377-2217. DOI: doi.org/10.1016/j.ejor.2021.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0377221721000084.

Reisch, Julian and Natalia Kliewer (2020). "Black-Box Optimization in Railway Simulations". In: *Operations Research Proceedings 2019*. Ed. by Janis S. Neufeld et al. Cham: Springer International Publishing, pp. 717–723. ISBN: 978-3-030-48439-2.

Reisch, Julian, Natalia Kliewer, et al. (2021). "Bestimmung der Kapazitätssteigerung durch Einführung der Mittelpufferkupplung und ep-Bremse (in German)". In: *ETR – Eisenbahntechnische Rundschau, Ausgabe 1-2/2021*.

Robertson, N. and P.D. Seymour (1995). "Graph Minors. XIII. The Disjoint Paths Problem". In: *Journal of Combinatorial Theory, Series B* 63.1, pp. 65–110. ISSN: 0095-8956. DOI: https://doi.org/10.1006/jctb.1995.1006. URL: http://www.science%20direct.com/science/article/pii/S0095895685710064.

Robertson, Neil and P.D Seymour (1984). "Graph minors. III. Planar tree-width". In: *Journal of Combinatorial Theory, Series B* 36.1, pp. 49–64. ISSN: 0095-8956. DOI: https://doi.org/10.1016/0095-8956(84)90013-3. URL: http://www.sciencedirect.com/science/article/pii/0095895684900133.

— (1991). "Graph minors. X. Obstructions to tree-decomposition". In: *Journal of Combinatorial Theory, Series B* 52.2, pp. 153–190. ISSN: 0095-8956. DOI: https://doi.org/10.1016/0095-8956(91)90061-N. URL: http://www.science%20direct.com/science/article/pii/009589569190061N.

Rosin, Christopher D. (2014). "Unweighted Stochastic Local Search can be Effective for Random CSP Benchmarks". In: *CoRR* abs/1411.7480. arXiv: 1411.7480. URL: http://arxiv.org/abs/1411.7480.

Rößler, David, Natalia Kliewer, and Julian Reisch (Feb. 2019). "Modeling delay propagation and transmission in railway networks". In: *Proceedings of the 14th International Conference on Wirtschaftsinformatik*. Vol. 14, pp. 98–111.

Rößler, David et al. (2021). "Discerning Primary and Secondary Delays in Railway Networks using Explainable AI". In: vol. 52. 23rd EURO Working Group on Transportation Meeting, EWGT 2020, 16-18 September 2020, Paphos, Cyprus, pp. 171–178. DOI: https://doi.org/10.1016/j.trpro.2021.01.018. URL: https://www.sciencedirect.com/science/article/pii/S2352146521000405.

Ruan, Jianhua, Angela K Dean, and Weixiong Zhang (2010). "A general co-expression network-based approach to gene expression analysis: comparison and applications". In: *BMC systems biology* 4.1, p. 8.

Rubin, Donald B. (1976). "Inference and missing data". In: *Biometrika* 63.3, pp. 581–592.

Schlaich, Johannes and Daniel Pöhle (2017). "Einsatz von Optimierungsverfahren in der Fahrplanerstellung (in German)". In: *Tagungsbericht Heureka 2017*.

Schöbel, Anita (2007). "Integer programming approaches for solving the delay management problem". In: *Algorithmic methods for railway optimization*. Springer, pp. 145–170.

— (2017). "An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation". In: *Transportation Research Part C: Emerging Technologies* 74, pp. 348–365. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2016.11.018. URL: http://www.sciencedirect.com/science/article/pii/S0968090X1630242X.

Schöbel, Anita and Albrecht Kratz (2009). "A Bicriteria Approach for Robust Timetabling". In: *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Ed. by Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 119–144. ISBN: 978-3-642-05465-5. DOI: 10.1007/978-3-642-05465-5_5. URL: https://doi.org/10.1007/978-3-642-05465-5_5.

Schoenberg, I. J. (1988). "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions". In: *I. J. Schoenberg Selected Papers*. Ed. by Carl de Boor. Boston, MA: Birkhäuser Boston, pp. 3–57. ISBN: 978-1-4899-0433-1. DOI: 10.1007/978-1-4899-0433-1_1. URL: https://doi.org/10.1007/978-1-4899-0433-1_1.

Schranil, Steffen (2013). *Prognose der Dauer von Störungen des Bahnbetriebs (in German)*. Vol. 164. ETH Zurich.

Schrijver, Alexander (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc. ISBN: 0471908541.

Schwarz, Adam J and John McGonigle (2011). "Negative edges and soft thresholding in complex network analysis of resting state functional connectivity data". In: *Neuroimage* 55.3, pp. 1132–1146. DOI: 10.1016/j.neuroimage.2010.12.047.

Selman, Bart, Henry Kautz, and Bram Cohen (1995). "Local Search Strategies for Satisfiability Testing". In: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 521–532.

Serafini, Paolo and Walter Ukovich (1989a). *A Mathematical Model for Periodic Scheduling Problems*. in: SIAM Journal on Discrete Mathematics, 2, pp. 550-581.

— (1989b). "A mathematical model for periodic scheduling problems". In: *SIAM Journal on Discrete Mathematics* 2.4, pp. 550–581.

Seymour, P. D. and R. Thomas (June 1994). "Call routing and the ratcatcher". English (US). In: *Combinatorica* 14.2, pp. 217–241. ISSN: 0209-9683. DOI: 10.1007/BF01215352.

Sloane, Neil J. A. (2015). *Challenge Problems: Independent Sets in Graphs*. URL: https://oeis.org/A265032/a265032.html.

Sutskever, Ilya et al. (17–19 Jun 2013). "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, pp. 1139–1147. URL: http://proceedings.mlr.press/v28/sutskever13.html.

Szmidt, Eulalia and Janusz Kacprzyk (2011). "The Spearman and Kendall rank correlation coefficients between intuitionistic fuzzy sets." In: *EUSFLAT Conf.* Pp. 521–528.

Tamaki, Hisao (May 2019). "Positive-Instance Driven Dynamic Programming for Treewidth". In: *J. Comb. Optim.* 37.4, pp. 1283–1311. ISSN: 1382-6905. DOI: 10.1007/s10878-018-0353-z. URL: https://doi.org/10.1007/s10878-018-0353-z.

Taylor, Richard (1990). "Interpretation of the correlation coefficient: a basic review". In: *Journal of diagnostic medical sonography* 6.1, pp. 35–39.

Thistlethwaite, D. L. and D. T. Campbell (1960). "Regression-discontinuity analysis: An alternative to the ex post facto experiment". In: *Journal of Educational Psychology* 51.6, pp. 309–317.

Tukey, John W (1977). *Exploratory data analysis*. Vol. 2. Reading, Mass.

Voth-Gaeddert, Lee and Devin Cornell (2016). "Improving health information systems in Guatemala using weighted correlation network analysis". In: *Global Humanitarian Technology Conference (GHTC), 2016*. IEEE, pp. 686–693.

Xu, Ke (2005). *BHOSLIB: Benchmarks with hidden optimum solutions for graph problems*. URL: http:%20//www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm.

Yuan, Jianxin (2006). *Stochastic modelling of train delays and delay propagation in stations*. Vol. 2006. Eburon Uitgeverij BV.

Zhang, Bin and Steve Horvath (2005). "A general framework for weighted gene coexpression network analysis". In: *Statistical applications in genetics and molecular biology* 4.1, pp. 1–45.

Zhang, Yongxiang et al. (2019). "Solving cyclic train timetabling problem through model reformulation: Extended time-space network construct and Alternating Direction Method of Multipliers methods". In: *Transportation Research Part B: Methodological* 128, pp. 344–379. ISSN: 0191-2615.

Zwaneveld, Peter J. et al. (1996). “Routing Trains Through Railway Stations: Model Formulation and Algorithms”. In: *Transportation Science* 30.3, pp. 181–194. ISSN: 1526-5447. DOI: 10.1287/trsc.30.3.181. URL: http://dx.doi.org/10.1287/trsc.30.3.181.