Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

NIELS LINDNER[1], CHRISTIAN LIEBCHEN[2]

Timetable Merging for the Periodic Event Scheduling Problem

[1] 0000-0002-8337-4387
[2] 0000-0002-4311-2024

ZIB Report 21-06 (February 2021)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30 84185-0
Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Timetable Merging for the Periodic Event Scheduling Problem

Niels Lindner [a,1], Christian Liebchen [b,2],
[a] Zuse Institute Berlin
Takustr. 7, 14195 Berlin, Germany
[1] E-mail: lindner@zib.de, Phone: +49 (0) 30 84185 374
[b] Technical University of Applied Sciences Wildau
Hochschulring 1, 15745 Wildau
[2] E-mail: liebchen@th-wildau.de

Abstract
We propose a new mixed integer programming based heuristic for computing new benchmark primal solutions for instances of the PESPlib. The PESPlib is a collection of instances for the Periodic Event Scheduling Problem (PESP), comprising periodic timetabling problems inspired by real-world railway timetabling settings, and attracting several international research teams during the last years. We describe two strategies to merge a set of good periodic timetables. These make use of the instance structure and minimum weight cycle bases, finally leading to restricted mixed integer programming formulations with tighter variable bounds. Implementing this timetable merging approach in a concurrent solver, we improve the objective values of the best known solutions for the smallest and largest PESPlib instances by 1.7 and 4.3 percent, respectively.

Keywords
Periodic Event Scheduling Problem, Periodic Timetabling, Railway Timetabling, PESPlib, Benchmark Solutions, Mixed Integer Programming

1 Introduction

In periodic event scheduling, we consider a set of events, where each event repeats after the very same period time T. A periodic timetable is then an assignment of time values within the interval $[0, T)$ to all the events, subject to the condition that the (periodic) time differences between certain pairs of events meet a feasibility interval.

This type of problem arises in particular in the context of computing cyclic timetables in public transport (aka fixed-interval timetables), where one may think of a periodic event as either arrival or departure of a directed traffic line at some station, and in coordinating the traffic lights at several road intersections (Hassin, 1996; Wünsch, 2008). In the context of timetabling in public transport, the Periodic Event Scheduling Problem (PESP) introduced by Serafini and Ukovich (1989) has been the core model for computing the first optimized railway timetable in practice Liebchen (2008), and also for the timetabling part of the company-wide success story of Operations Research at Dutch Railways (Kroon et al., 2009).

In order to attract researchers in developing more powerful solution methods for periodic event scheduling, Goerigk (2012) composed a collection of instances (PESPlib), most of which are motivated from railway timetabling. This library attracted several international research teams, however, none of the PESPlib instances has been solved to proven optimality. The current incumbent solutions stem from the concurrent PESP solver by Borndörfer et al. (2020), in which a variety of algorithms and heuristics are used as subroutines. In particular, the most common local search procedures are implemented, such that the best solutions constitute local minima for *all* of these heuristics.

We present a strategy to escape local minima. We follow the spirit of by Cook and Seymour (2003): Their *tour merging heuristic* constructs the union of ten heuristically determined tours for the Traveling Salesman Problem (TSP), applies a branch-decomposition-based dynamic program to this restricted instance, and is able to produce even better TSP tours.

Our philosophy is to *merge* a small set of good quality timetables. But we do not copy two half timetables and simply compile their parts. Rather, we do so by restricting the bounds of the periodic tension or cycle offset variables in the cycle-based mixed integer programming formulation of PESP. We exploit the similarities of the input timetables, and take into account the special characteristics of a periodic timetabling problem in public transport. This leads to two timetable merging heuristics, one for each variable type.

From the perspective of mixed integer programs, our method can be understood as a generalization of the well-known crossover heuristic (Rothberg, 2007). We tighten the variable bounds such that *all* the solutions in the selected small set of good quality timetables remain feasible. However, our approach is more general, as we consider more than two initial solutions and allow bound restrictions of arbitrary variables instead of merely fixing integer variables with the same solution value. Moreover, the crossover heuristic for general mixed integer programs is supposed to run very fast for the purpose of solution polishing, but we spend a considerable amount of time on solving the restricted problem, as solution quality is our main goal, and PESP is notoriously hard.

It turns out that applying this tailored timetable merging heuristic can make better solutions accessible. Invoking the concurrent PESP solver that computed the current PESPlib incumbents, we find several timetables improving the best known primal bounds, and end up with a 1.7 and 4.3 percent lower objective value for the smallest and largest PESPlib instance, respectively.

We define PESP in Section 2, where we also review some problem features, and introduce the cycle-based mixed integer programming formulation. Our timetable merging heuristic is discussed in Section 3. Section 4 presents computational results, on the one hand on the structure of the restricted scenarios from the merging heuristic, and on finding better periodic timetables for the PESPlib instances on the other. We conclude this paper in Section 5.

2 The Periodic Event Scheduling Problem

In this section we first define the combinatorial optimization problem for which we are about to propose our merging heuristics, provide a short review of other solution techniques that have been applied earlier, and on some of which we are going to build on, and formulate the mixed-integer linear programming (MIP) formulation that constitutes the basis of our computational study.

2.1 Problem Definition

The *Periodic Event Scheduling Problem* (PESP) dates back to Serafini and Ukovich (1989), and shows certain similarities to models that were already considered by Rüger (1986). Formally, the input is a tuple (G, T, ℓ, u, w), where

- $G = (V, A)$ is a directed graph, called *event-activity network*, whose vertices are called *events* and whose arcs are called *activities*,
- $T \in \mathbb{N}$ is a *period time*,
- $\ell \in \mathbb{R}^A_{\geq 0}$ is a vector of *lower bounds* such that $0 \leq \ell < T$,
- $u \in \mathbb{R}^A_{\geq 0}$ is a vector of *upper bounds*, $0 \leq u - \ell < T$, and
- $w \in \mathbb{R}^A_{\geq 0}$ is a vector of *weights*.

A vector $\pi \in [0, T)^V$ is called a *periodic timetable* if there exists a *periodic tension* $x \in \mathbb{R}^A$ such that

$$\ell \leq x \leq u \quad \text{and} \quad \forall a = (i, j) \in A : \ \pi_j - \pi_i \equiv x_a \bmod T.$$

Intuitively, a periodic timetable π assigns times modulo T to each event in G, and fixes the duration of any activity $a = (i, j) \in A$ to $\pi_j - \pi_i$ modulo T. The actual duration of a is then chosen to lie within the interval $[\ell_a, u_a]$. Starting from a periodic timetable π, a corresponding periodic tension x can be computed by

$$x_a := [\pi_j - \pi_i - \ell_a]_T + \ell_a \quad \text{for all } a = (i, j) \in A,$$

where $[\cdot]_T$ denotes the modulo T operator with values in $[0, T)$. Conversely, given a periodic tension x, a corresponding periodic timetable π can be reconstructed by a graph traversal. We further define the *periodic slack* as $y := x - \ell \in \mathbb{R}^A_{\geq 0}$.

In a public transport context, an event i is usually modeling either the arrival or the departure of a directed traffic line at some station, e.g., the departure of the trains from Berlin to Munich in the city of Erfurt. An arc $a = (i, j)$ models the time duration from event i to event j. If i and j are two subsequent departure and arrival events of the same directed line, then $a = (i, j)$ models the trip duration from the station of event i to the station of event j. In turn, if i and j are the arrival and departure events of the same directed line within the same station, then $a = (i, j)$ models the dwell duration within this station. To illustrate many other commercial and operational types of constraints, we refer to Liebchen and Möhring (2004). If in an hourly service (i.e., $T = 60$ minutes), for a dwell arc $a = (i, j)$ we require that $\ell_a = 3$ and $u_a = 7$, then of course $\pi_i = 29$ and $\pi_j = 33$ constitute a feasible timetable, because $33 - 29 = 4 \in [3, 7]$. However, notice that $\pi_i = 58$ and $\pi_j = 3$ constitute a feasible timetable, too, because $x_a = [3 - 58 - 3]_{60} + 3 = 2 + 3 = 5 \in [3, 7]$.

Definition 2.1. Given (G, T, ℓ, u, w) as above, the *Periodic Event Scheduling Problem* (PESP) is to find a periodic slack y such that the weighted periodic slack $\sum_{a \in A} w_a y_a$ is minimum or to decide that no periodic timetable exists.

Alternatively, one may seek to minimize the weighted periodic tension $\sum_{a \in A} w_a x_a$, which differs from the corresponding weighted periodic slack by the constant $\sum_{a \in A} w_a \ell_a$. Speaking in application terms, where weights may reflect the number of passengers using

an activity, this amounts to minimizing the total passenger travel time, given that the routes that the passengers are taking – and thus the activities on which they are showing up – are known in advance. Notice that recently there have been made advances in relaxing this assumption (Schiewe and Schöbel, 2020). But the activities' weights may also model other practical aspects, as for instance in the context of minimizing the amount of rolling stock that is required to operate a periodic timetable (Liebchen and Möhring, 2004).

Notice that in the literature, sometimes PESP is regarded as the pure feasibility decision problem, thus with no weights w_a defined on the arcs.

2.2 Complexity and Solution Approaches

For an arbitrary PESP instance, it is not clear at all that a periodic timetable resp. tension resp. slack exists. In fact, the feasibility problem is NP-complete even if $T \geq 3$ is not regarded as part of the input, because PESP generalizes Vertex Coloring (Odijk, 1994). Moreover, the feasibility can be checked in linear time if undirecting G results in a forest, but is already NP-hard for graphs of treewidth or branchwidth ≥ 2 (Lindner and Reisch, 2020).

In particular, in this paper we deal with an NP-hard optimization problem. A lot of powerful tools stimulated by different viewpoints within discrete optimization have been developed to solve PESP instances: These tools comprise, e.g., mixed-integer programming (Liebchen, 2006), simplex-style algorithms (Nachtigall and Opitz, 2008; Goerigk and Schöbel, 2013), satisfiability methods (Großmann et al., 2012), machine learning (Matos et al., 2020), matching heuristics (Pätzold and Schöbel, 2016), and graph partitioning approaches (Lindner and Liebchen, 2019). A large part of these methods has been integrated into a single PESP solver based on concurrency, `ConcurrentPESP` (Borndörfer et al., 2020).

However, up to today even medium-sized PESP instances withstand all attempts to compute optimal solutions: Since 2012, none of the 20 instances of the PESP benchmark library PESPlib (Goerigk, 2012) could be solved to optimality, the current best primal-dual gap being $\approx 35\%$, where the smallest instance R1L1 even has no more than 6,386 activities. For all these instances, `ConcurrentPESP` is the record holder concerning both primal and dual bounds.

2.3 Mixed Integer Programming Formulation

Our merging strategy relies on the well-known *cycle-based mixed integer programming formulation* for PESP:

$$\begin{aligned} \text{Minimize} \quad & \sum_{a \in A} w_a (x_a - \ell_a) \\ \text{subject to} \quad & \Gamma x = Tz, \\ & \ell \leq x \leq u, \\ & z \in \mathbb{Z}^B. \end{aligned} \tag{1}$$

Here, B denotes an integral cycle basis of G with cycle matrix Γ, see, e.g., Liebchen (2006) for details. The variables are the periodic tension x as defined above and the *cycle offset* $z \in \mathbb{Z}^B$. Starting from a periodic tension x, the corresponding cycle offset is of course

recovered by computing $z = \Gamma x/T$. Conversely, given z, an optimal periodic tension x for z can be found by a minimum cost network flow computation (Nachtigall and Opitz, 2008). In the case of integer input vectors ℓ and u, the existence of a feasible solution implies the existence of an integral solution vector x, as Γ is the cycle matrix of an integral cycle basis.

In addition, recall from (Odijk, 1994) that for each oriented cycle $\gamma \in \{-1, 0, 1\}^A$ the following *cycle inequalities*

$$\underline{z}_\gamma^{\text{ODIJK}} := \left\lceil \frac{\gamma_+^t \ell - \gamma_-^t u}{T} \right\rceil \leq z_\gamma \leq \left\lfloor \frac{\gamma_+^t u - \gamma_-^t \ell}{T} \right\rfloor =: \overline{z}_\gamma^{\text{ODIJK}} \tag{2}$$

are valid and turned out to be useful for solving (1) in many computational studies. Here, $\gamma_+ := \max(\gamma, 0)$ resp. $\gamma_- := \max(-\gamma, 0)$ denote the positive resp. negative part of γ, so that $\gamma = \gamma_+ - \gamma_-$.

3 Merging Timetables

The incumbent solutions in the PESPlib are hard to improve further: They are locally optimal for the local heuristics implemented in the `ConcurrentPESP` solver, so that only the global component realized by branch-and-cut is able to yield better solutions. This is unsatisfactory, given the enormous size of the branch-and-bound trees being a result of the weak trivial linear programming relaxations (Liebchen, 2006).

Bound restriction approaches are not new to periodic event scheduling. For example, the *Arc Selection* heuristic suggested by Roth (2019) tries to improve a periodic tension x by decreasing the upper bounds u to x and iteratively re-optimizing over subsets of activities. The optimization is carried out by calling a MaxSAT solver after transforming the PESP instance to an instance of the weighted partial maximum satisfiability problem. On large instances, this approach is reported to be superior to pure branch-and-cut.

Although inspired by Roth (2019), our idea is somewhat different: For a PESP instance, we consider a set S of solutions with "good" objective values. We now want to identify certain variables of (1) whose values are the same or at least similar for all the solutions in S. Then, by sharpening bounds for these variables, we aim at deriving a modified mixed-integer program that

- is easier to solve because of the tighter bounds,
- still contains all the already good solutions of S as feasible solutions, and
- has a feasible set which is a subset of the feasible set of the initial problem.

Ideally, solving this sharpened problem, we could find even better solutions for the initial problem. As there are two types of variables in (1), there are two directions to pursue: Restricting the periodic tension x or restricting the cycle offset z. We will discuss the details of our strategy for these two cases separately.

3.1 Restricting Periodic Tensions

Given a PESP instance $I = (G, T, \ell, u, w)$, we first analyze the *span* $u_a - \ell_a$ of an activity a. For example, in the smallest PESPlib instance R1L1, having $T = 60$, it turns out that each activity either has span at most 17 or exactly 59. A similar structure is present in the

other railway instances of the PESPlib. Motivated by the typical modeling of event-activity networks for periodic timetabling in public transport, we refer to these *free* activities in the group with the larger span as *transfer activities*, which is also supported by an investigation of the connectivity structure of the PESPlib instances due to Goerigk and Liebchen (2017). With integer bounds, as is the case for the PESPlib instances, a span of 59 on an activity a in conjunction with a period time $T = 60$ means that any integer periodic slack $y_a \in [0, 60)$ is feasible. However, restricting the span of these activities in a too rigorous way might turn the PESP instance infeasible.

Of course, increasing the lower bounds and decreasing the upper bounds of the activities is our main goal. However, in order to keep some flexibility, we restrict the upper bounds in a different manner than the lower bounds: Non-transfer activities as well as transfer activities with low impact on the objective function always remain at their original upper bounds, because there would be no strong empirical evidence which of these arcs will have to face large slack in the best solutions for an instance. To this end, we choose a parameter $\alpha \in [0, 1]$ and denote by A_α the largest set of transfer activities $a \in A$ that, when sorted in ascending order w.r.t. weight w_a, sum up to at most $\alpha \sum_{a \in A} w_a$.

Now let S be a set of periodic tensions for I. We define the following bounds for each activity $a \in A$:

$$\ell_a^{\text{ORIG}} := \ell_a, \qquad u_a^{\text{ORIG}} := u_a,$$

$$\ell_a^{\text{FIX}} := \min\{x_a \mid x \in S\}, \qquad u_a^{\text{FIX}} := \begin{cases} \max\{x_a \mid x \in S\} & \text{if } a \in A_\alpha, \\ u_a & \text{otherwise,} \end{cases}$$

$$\ell_a^{\text{MED}} := \frac{1}{2}\ell_a^{\text{ORIG}} + \frac{1}{2}\ell_a^{\text{FIX}}, \qquad u_a^{\text{MED}} := \frac{1}{4}u_a^{\text{ORIG}} + \frac{3}{4}u_a^{\text{FIX}}.$$

Observe that for all $x \in S$ we find

$$\ell = \ell^{\text{ORIG}} \leq \ell^{\text{MED}} \leq \ell^{\text{FIX}} \leq x \leq u^{\text{FIX}} \leq u^{\text{MED}} \leq u^{\text{ORIG}} = u.$$

We can now combine different lower and upper bound strategies freely by considering the PESP instances $I_S(L, U) := (G, T, \ell^L, u^U, w)$ for $L, U \in \{\text{ORIG}, \text{MED}, \text{FIX}\}$. Clearly, $I_S(\text{ORIG}, \text{ORIG})$ is the original PESP instance I. Observe that any solution in S is feasible for all scenarios $I_S(L, U)$. Moreover, restricting the changes to either lower or upper bound, a periodic tension for FIX is a periodic tension for MED, and any periodic tension for MED is a periodic tension for ORIG. However, formally, the weighted slacks might differ, as the lower bounds are potentially altered.

3.2 Restricting Cycle Offsets

Consider a PESP instance $I = (G, T, \ell, u, w)$ with a set S of periodic tensions. For every integral cycle basis B with cycle matrix Γ, we can compute the corresponding cycle offset vector $z \in \mathbb{Z}^B$ for each tension $x \in S$. Denote by $S_B := \{\Gamma x/T \mid x \in S\}$ the set of these cycle offsets. The idea is now to find bounds $\underline{z}, \overline{z} \in \mathbb{Z}^B$ and to solve the restricted mixed-integer program

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{a \in A} w_a y_a \\
\text{subject to} \quad & \Gamma x = T z, \\
& \ell \leq x \leq u, \\
& \underline{z} \leq z \leq \overline{z}, \\
& z \in \mathbb{Z}^B.
\end{aligned} \tag{3}
$$

We do this in two ways by defining for each oriented cycle $\gamma \in B$

$$
\begin{aligned}
\underline{z}_\gamma^{\text{ALL}} &:= \min\{z_\gamma \mid z \in S_B\}, \\
\overline{z}_\gamma^{\text{ALL}} &:= \max\{z_\gamma \mid z \in S_B\}, \\
\underline{z}_\gamma^{\text{PARTIAL}} &:= \begin{cases} \underline{z}_\gamma^{\text{ALL}} & \text{if } \underline{z}_\gamma^{\text{ALL}} = \overline{z}_\gamma^{\text{ALL}}, \\ \underline{z}_\gamma^{\text{ODIJK}} & \text{otherwise, cf. (2)}, \end{cases} \\
\overline{z}_\gamma^{\text{PARTIAL}} &:= \begin{cases} \overline{z}_\gamma^{\text{ALL}} & \text{if } \underline{z}_\gamma^{\text{ALL}} = \overline{z}_\gamma^{\text{ALL}}, \\ \overline{z}_\gamma^{\text{ODIJK}} & \text{otherwise, cf. (2)}. \end{cases}
\end{aligned}
$$

In the PARTIAL version, we fix all cycle offset variables if they are the same in all solutions in S, and we do not impose any restrictions that would depend on S on the other entries of z.

Hence, for any set of tensions S and any integral cycle basis B, we obtain two restricted mixed-integer programs of the form (3): $\text{MIP}_{S,B}(I, \text{ALL})$ and $\text{MIP}_{S,B}(I, \text{PARTIAL})$. Note that these do not necessarily correspond to PESP instances anymore, as we have not changed any part of the input data $I = (G, T, \ell, u, w)$ – we are merely restricting the program (1). Clearly, the solutions in S are feasible for both $\text{MIP}_{S,B}(I, \text{ALL})$ and $\text{MIP}_{S,B}(I, \text{PARTIAL})$. Moreover, any feasible solution to $\text{MIP}_{S,B}(I, \text{ALL})$ is feasible for $\text{MIP}_{S,B}(I, \text{PARTIAL})$, and any feasible solution to $\text{MIP}_{S,B}(I, \text{PARTIAL})$ is feasible for the original unrestricted MIP (1). As the cycle inequalities (2) are valid inequalities, the unrestricted MIP (1) is equivalent to $\text{MIP}_{S,B}(I, \text{ODIJK})$ for any cycle basis B and any S. Referring to $\text{MIP}_{S,B}(I, \cdot)$ as the feasible regions of the respective mathematical programs, we summarize

$$
S \subseteq \text{MIP}_{S,B}(I, \text{ALL}) \subseteq \text{MIP}_{S,B}(I, \text{PARTIAL}) \subseteq \text{MIP}_{S,B}(I, \text{ODIJK}) \equiv \text{MIP (1)}.
$$

3.3 Choosing a Cycle Basis

So far, we have not discussed which cycle basis B to choose when restricting cycle offsets. For the purpose of solving the mixed integer program (3), it is desirable to fix as many integer variables as possible, and more generally, to minimize the possible number of values of z. Hence, we seek to minimize

$$
\prod_{\gamma \in B} (\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} + 1),
$$

or equivalently,

$$
\log \prod_{\gamma \in B} (\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} + 1) = \sum_{\gamma \in B} \log(\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} + 1). \tag{4}
$$

We call (4) the *log width* of the cycle basis B.

In the context of periodic timetabling, the concept of integral cycle bases of small (log) width has proven to be a valuable tool in order to speed up the branch-and-bound process for the cycle-based mixed integer programming formulation of PESP (Liebchen and Peeters, 2009). Finding a so-called directed or undirected cycle basis of minimum weight is well-understood (Kavitha et al., 2009).

Yet, in order to profit from these insights, we must get around the following two pitfalls: First, since a minimum-weight cycle basis in general is not integral, after having computed a minimum-weight cycle basis we need to check, whether it is even an integral cycle basis, because otherwise the computed cycle basis risks to be useless. Second, we can only compute a minimum-weight cycle basis efficiently, if the weight is given as a function on the arcs rather than on the cycles. Therefore, we will construct an arc-weight function $c : A \to \mathbb{R}_{\geq 0}$ such that for all oriented cycles γ in G holds

$$c(\gamma) := \sum_{a \in \gamma} c(a) \approx \overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}}.$$

Lemma 3.1. *Define $c : A \to \mathbb{R}_{\geq 0}$ via*

$$c(a) := \frac{\max\{x_a \mid x \in S\} - \min\{x_a \mid x \in S\}}{T}.$$

Then for all oriented cycles $\gamma \in B$ consisting of $|\gamma|$ arcs holds

$$0 \leq c(\gamma) - (\overline{z}_\gamma^{ALL} - \underline{z}_\gamma^{ALL}) \leq c(\gamma) < |\gamma|.$$

Proof. Let γ be an oriented cycle in B. Then

$$\begin{aligned}
\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} &= \max\{z_\gamma \mid z \in S_B\} - \min\{z_\gamma \mid z \in S_B\} \\
&= \frac{\max\{(\Gamma x)_\gamma \mid x \in S\} - \min\{(\Gamma x)_\gamma \mid x \in S\}}{T} \\
&= \frac{\max\{\gamma^t x \mid x \in S\} - \min\{\gamma^t x \mid x \in S\}}{T}.
\end{aligned}$$

Recall that we can decompose $\gamma = \gamma_+ - \gamma_-$ into its positive and negative part. Then for any $x \in S$ holds

$$\gamma_+^t \underline{x} - \gamma_-^t \overline{x} \leq \gamma^t x \leq \gamma_+^t \overline{x} - \gamma_-^t \underline{x},$$

where $\overline{x}, \underline{x}$ are the vectors in $\mathbb{R}^A$ with

$$\overline{x}_a := \max\{x_a \mid x \in S\} \quad \text{and} \quad \underline{x}_a := \min\{x_a \mid x \in S\} \quad \text{for all } a \in A.$$

In particular

$$\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} \leq \frac{(\gamma_+^t \overline{x} - \gamma_-^t \underline{x}) - (\gamma_+^t \underline{x} - \gamma_-^t \overline{x})}{T} = \frac{(\gamma_+ + \gamma_-)^t (\overline{x} - \underline{x})}{T} = \sum_{a \in \gamma} c(a) = c(\gamma).$$

Since $\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} \geq 0$, we arrive at

$$0 \leq c(\gamma) - (\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}}) \leq c(\gamma).$$

Finally note that, as we require $u - \ell < T$ for our PESP instances, we have

$$c(\gamma) \leq \sum_{a \in \gamma} \frac{u_a - \ell_a}{T} < |\gamma|. \qquad \square$$

We can now compute a minimum-weight undirected cycle basis B^* w.r.t. c. Note that such a cycle basis is not necessarily integral. However, our empirical observation is that on non-artificial PESP instances, as the ones that can be found in particular in the PESPlib, minimum undirected cycle bases typically turn out to be integral. Being an $\mathbb{F}_2$-vector space, the undirected cycles form a matroid (Horton, 1987), so that B^* is also of minimum weight w.r.t. $\log(c)+1$: Up to breaking ties, the greedy algorithm sorts the cycles in the same order both regarding c and $\log(c)+1$. This enables us to bound the error that we make when we are using the cycle basis B^* instead of the one that minimizes the actual log width (4).

Corollary 3.2. *Let B' be an undirected cycle basis of G of minimum log width* (4)*. Then*

$$0 \leq \sum_{\gamma \in B^*} \log(\overline{z}_\gamma^{ALL} - \underline{z}_\gamma^{ALL} + 1) - \sum_{\gamma \in B'} \log(\overline{z}_\gamma^{ALL} - \underline{z}_\gamma^{ALL} + 1) < |B^*| \log(|V| + 1).$$

Proof. The left inequality is clear as B' minimizes (4). As $\sum_{\gamma \in B'} \log(\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} + 1) \geq 0$, it remains to invoke Lemma 3.1 to obtain

$$\sum_{\gamma \in B^*} \log(\overline{z}_\gamma^{\text{ALL}} - \underline{z}_\gamma^{\text{ALL}} + 1) < \sum_{\gamma \in B^*} \log(|\gamma| + 1) \leq |B^*| \log(|V| + 1).$$

Note that we can assume $|\gamma| \leq |V|$ as there is always a minimum undirected cycle basis composed of simple cycles, and we only compare the objectives. □

An empirical analysis of the quality of different cycle bases will be given in §4.2, the cycle basis B^* as defined in Lemma 3.1 being superior for our purposes.

4 Results

We turn now to the computational results that we obtained by timetable merging. We describe the details of the set-up in §4.1. In §4.2, we present a structural analysis of the restricted scenarios constructed in §3, including an assessment of various cycle bases. Finally, the actual timetables found by our strategy are presented in §4.3.

4.1 General Setup

We test the merging approach outlined in Section 3 on the smallest and largest PESPlib instance, R1L1 and R4L4, respectively. For each of the two instances, we choose a set S of 5 solutions whose weighted slack is close to the current PESPlib record. Among these solutions are the incumbent solution found by Lindner and Roth (see Borndörfer et al. 2020), and the best solution found by the iterative procedure by Goerigk and Liebchen (2017).

Our workflow is visualized in Figure 1. We construct the 8 scenarios with the tightened bounds as described in §3.1, and another 2 scenarios according to §3.2. For each of these 10 scenarios, we conduct $18 = 6 \cdot 3$ runs of `ConcurrentPESP` with 60 minutes each (wall time): These arise from 6 different initial solutions (the ones from the set S, or none) in combination with 3 different parameter settings for fine-tuning the actual solution process (Round 1). For each of these runs we check whether the optimum solution of that run achieved an improvement compared to the initial solution that gave rise to the respective run. If this is the case, then the possibility arises to escape local minima and we thus launch Round 2: We feed the `ConcurrentPESP` solver on the original instance R1L1 resp. R4L4

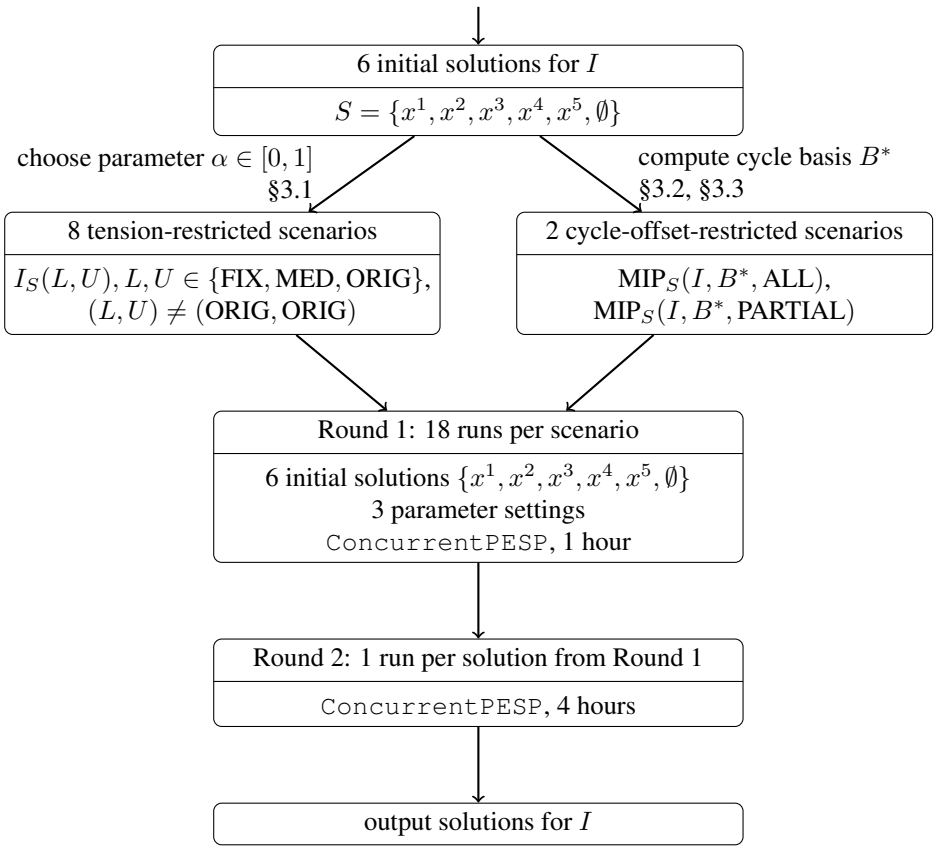

Figure 1: Workflow for a PESP instance I

with the output of Round 1 as input. Each such run of Round 2 is performed as a single solver run for 4 hours (wall time), dedicating most computational capacity to MIP solving.

For the scenarios of Round 1, we compute dual bounds with separated additional dedicated runs of `ConcurrentPESP` of 24 hours each, running exclusively a MIP solver with best bound emphasis, and a separator for violated flip inequalities (Lindner and Liebchen, 2020). For the ALL and PARTIAL scenarios with restricted cycle offsets, we have to turn off the modulo network simplex algorithm, as this too frequently violates the tighter bounds. Instead, we adjust the mixed-integer programming based maximum cut heuristic available in `ConcurrentPESP` to work with the cycle offset bound constraints.

The `ConcurrentPESP` solver is run on up to 8 threads on an Intel Xeon E3-1270 CPU running at 3.80 GHz with 32 GB RAM. We use IBM CPLEX 12.10 as underlying MIP solver.

4.2 Scenario Analysis

Tension-Restricted Scenarios

The approach from §3.1 yields PESP instances $I_S(L, U)$ for $L, U \in \{\text{FIX}, \text{MED}, \text{ORIG}\}$, where $I_S(\text{ORIG}, \text{ORIG})$ corresponds to the original instance $I \in \{\text{R1L1}, \text{R4L4}\}$, and is thus omitted. Our approach does neither alter the graph G nor the weights w, it merely affects the lower bounds ℓ and the upper bounds u. We demonstrate hence the effect of our approach by analyzing the *span*: The span of an activity $a \in A$ is defined as $u_a - \ell_a$. We will call an activity *fixed* if its span is 0, and *free* if the span is 59. The *weighted span* of an activity a is given by $w_a(u_a - \ell_a)$.

Table 1 presents an analysis of the activity spans in the tension-restricted scenarios for the PESPlib instances R1L1 and R4L4. As parameter α, we chose $\alpha = 0.125$.

The "difficulty" of the restricted instances $I(L, U)$, in terms of the average span or the average weighted span, is dictated by the upper bound strategy: $U = \text{FIX}$ has lower average span than $U = \text{MED}$, which in turn has lower average span than $U = \text{ORIG}$. Each change in U to a more restrictive policy reduces the weighted average span by roughly a factor of $\frac{1}{2}$. For a given strategy for the upper bounds, there is a clear ranking for the lower bound strategies, which is also FIX $\rightarrow$ MED $\rightarrow$ ORIG with increasing difficulty. Moreover, the number of free activities is significantly lower than on the original instance. The number of fixed activities is approximately the same for all scenarios with $U \in \{\text{MED}, \text{ORIG}\}$, but jumps up to $\approx$ 54-55 % (R1L1) resp. $\approx$ 43 % (R4L4) when $U = \text{FIX}$.

I	L	U	fixed activities	free activities	avg. span	avg. wt. span
R1L1	FIX	FIX	55.18 %	28.03 %	18.98	6 511
	MED	FIX	53.78 %	28.05 %	19.76	7 608
	ORIG	FIX	53.78 %	28.07 %	20.49	8 646
	FIX	MED	10.34 %	29.13 %	21.03	15 285
	MED	MED	10.12 %	29.24 %	21.81	16 382
	ORIG	MED	10.12 %	29.79 %	22.55	17 420
	FIX	ORIG	10.34 %	35.07 %	26.19	35 391
	MED	ORIG	10.12 %	35.61 %	26.96	36 488
	ORIG	ORIG	10.12 %	44.28 %	27.70	37 526
R4L4	FIX	FIX	42.78 %	32.38 %	24.99	3 802
	MED	FIX	42.66 %	32.40 %	25.76	4 073
	ORIG	FIX	42.66 %	32.54 %	26.46	4 320
	FIX	MED	8.93 %	34.36 %	27.02	7 643
	MED	MED	8.86 %	34.70 %	27.79	7 915
	ORIG	MED	8.86 %	36.53 %	28.50	8 162
	FIX	ORIG	8.93 %	40.91 %	31.84	16 221
	MED	ORIG	8.86 %	42.01 %	32.61	16 493
	ORIG	ORIG	8.86 %	54.27 %	33.32	16 740

Table 1: Span analysis of the tension-restricted R1L1 and R4L4 scenarios

Cycle-Offset-Restricted Scenarios and Evaluation of Cycle Bases

Recall from §3.3 that, when restricting the cycle offset vectors z, we want to choose a cycle basis B that minimizes the log width (4). In a kind of pre-test, we compare several cycle

bases:

1. a fundamental cycle basis obtained from a minimum spanning tree in G w.r.t. w,
2. an undirected cycle basis minimizing the span $u - \ell$,
3. an undirected cycle basis minimizing the number of arcs,
4. an undirected cycle basis minimizing the function c as defined in Lemma 3.1.

For our two instances, all the undirected cycle bases turn out to be integral, so that they are minimum integral cycle bases as well.

Table 2 evaluates the fixed cycle offset variables and the log width of the aforementioned cycle bases. The cycle basis $B^\star$ minimizing c suggested in §3.3 comes out as a clear winner: It fixes by far the most variables and the log width is smallest for our strategies ALL and PARTIAL. It is clear by construction that the strategies ALL and PARTIAL fix the same set of cycle offset variables. However, it is remarkable that none of the cycle bases is able to fix an integer variable in the standard MIP formulation (1) just with Odijk's cycle inequalities (2). This is due to the fact that every cycle in the R1L1 resp. R4L4 instance contains at least two transfer activities. The minimum span cycle basis is tailored to decrease the log width w.r.t. the Odijk bounds (Liebchen and Peeters, 2009), but surprisingly, minimizing the number of activities in the cycle basis often produces an even smaller log width. Minimizing c is a bad choice for tightening Odijk's bounds.

		ALL		PARTIAL		ODIJK	
I	cycle basis	fixed	log width	fixed	log width	fixed	log width
R1L1	fundamental	33.28 %	667	33.28 %	1 516	0.00 %	2 107
	min span	58.67 %	343	58.67 %	645	0.00 %	1 543
	min # arcs	59.59 %	346	59.59 %	609	0.00 %	1 484
	min c	73.62 %	216	73.62 %	597	0.00 %	2 302
R4L4	fundamental	22.01 %	2 862	22.01 %	6 542	0.00 %	7 939
	min span	37.66 %	1 777	37.66 %	3 399	0.00 %	5 378
	min # arcs	42.26 %	1 722	42.26 %	2 932	0.00 %	5 000
	min c	59.46 %	1 144	59.46 %	2 886	0.00 %	6 957

Table 2: Evaluation of cycle bases for R1L1 and R4L4. The "fixed" column indicates the percentage of cycles γ in the basis with $\underline{z}_\gamma = \overline{z}_\gamma$. The log width is given w.r.t. decadic logarithms to make the numbers more intuitive.

In the sequel, we thus select $\text{MIP}_{S,B^*}(\text{R1L1}, \text{ALL})$ and $\text{MIP}_{S,B^*}(\text{R1L1}, \text{PARTIAL})$ for our computations with the two cycle-offset-restricted scenarios.

4.3 Computational Results

We finally turn to the timetables found by our merging approach. In this section, we omit the subscripts S and B^*, as we consider only a single S and a single B^* per instance.

The results for R1L1 are summarized in Table 3 (Round 1) and Table 4 (Round 2). In Round 1, R1L1(ORIG, MED) is the only scenario where we could produce a better timetable (best weighted slack: 30 036 475) than the current PESPlib incumbent (weighted

slack: 30 415 672). This underlines the "hardness" of the local optimum at the latter solution. The scenarios R1L1$(-, \text{FIX})$ can in fact be solved to optimality: The current PESPlib incumbent is optimal. In particular, we cannot gain much information out of these 3 scenarios and discard them for Round 2. However, when reaching the computation time limit, several runs of Round 1 end up at a different timetable than the current PESPlib incumbent with only slightly higher slack. These timetables turn out to be "far enough" from the incumbent, and this is why we find improving solutions for all scenarios except $U = \text{FIX}$ in Round 2.

The best timetable has weighted slack 29 894 745 and is computed from one of the three better timetables produced by R1L1$(\text{ORIG}, \text{MED})$ in Round 1, but not from the one with lowest weighted slack. The PARTIAL cycle offset strategy produces the second best timetable with a weighted slack of 29 907 781.

L	U	best objective	better objectives	dual bound	gap
FIX	FIX	30 415 672	0 / 18	30 415 672	0.00 %
MED	FIX	30 415 672	0 / 18	30 415 672	0.00 %
ORIG	FIX	30 415 672	0 / 18	30 415 672	0.00 %
FIX	MED	30 415 672	0 / 18	29 673 500	2.44 %
MED	MED	30 415 672	0 / 18	28 612 511	5.93 %
ORIG	MED	30 036 475	3 / 18	25 883 704	13.83 %
FIX	ORIG	30 415 672	0 / 18	28 697 443	5.65 %
MED	ORIG	30 415 672	0 / 18	25 620 519	15.77 %

cycle offset strategy	best objective	better objectives	dual bound	gap
ALL	30 415 672	0 / 18	30 174 317	0.79 %
PARTIAL	30 415 672	0 / 18	30 118 941	0.98 %

Table 3: Results for the $10 = 8 + 2$ scenarios of Round 1 for R1L1

L	U	best objective	better objectives
FIX	MED	30 348 574	2 / 18
MED	MED	30 003 486	4 / 18
ORIG	MED	**29 894 745**	6 / 18
FIX	ORIG	30 373 924	1 / 18
MED	ORIG	30 335 565	3 / 18

cycle offset strategy	best objective	better objectives
ALL	29 973 362	7 / 18
PARTIAL	29 907 781	6 / 18

Table 4: Results of Round 2 for R1L1

The evolution of the objective values over both rounds for all 18 runs of the scenarios R1L1$(\text{ORIG}, \text{MED})$ and MIP$(\text{R1L1}, \text{ALL})$ is visualized in Figure 2 and Figure 3, respectively. These figures, as well as the subsequent ones, read as follows: The upper end of the green bar is the objective value of the initial solution from S that Round 1 has been fed with, or ∞ in the case where no solution has been provided to Round 1 ($\emptyset$). The border between

the green and the yellow bars is the objective value of the best solution that has been found in Round 1, and is thus plugged into Round 2 as initial solution. Finally, the bottom of the yellow bar is the objective value that has been achieved as the result of Round 2.

The picture for R4L4 is somewhat different: Although the current PESPlib incumbent is the optimal solution to the R4L4$(-, \text{FIX})$ instances, we find at least 6 better timetables for each other scenario in Round 1, the best has weighted slack 37 281 703 and is found by R4L4$(\text{ORIG}, \text{MED})$, too.

For Round 2, we again discard $U = \text{FIX}$. The remaining 7 scenarios bring plenty of better solutions, at least 12 per scenario. The best timetable is an outcome of a Round 1 timetable for R4L4$(\text{MED}, \text{ORIG})$, and has weighted slack 36 729 402. The objective values of the best timetables found in Round 2 are close for all scenarios, the second best with a weighted slack of 36 753 295 has again been produced by the PARTIAL cycle offset strategy. The objective value evolution for R4L4$(\text{MED}, \text{ORIG})$ and MIP$(\text{R4L4}, \text{PARTIAL})$ is visualized in Figure 4 and Figure 5, respectively.

L	U	best objective	better objectives	dual bound	gap
FIX	FIX	38 381 922	0 / 18	38 381 922	0.00 %
MED	FIX	38 381 922	0 / 18	38 381 922	0.00 %
ORIG	FIX	38 381 922	0 / 18	38 381 922	0.00 %
FIX	MED	38 095 741	7 / 18	29 918 556	21.46 %
MED	MED	37 616 308	6 / 18	24 688 127	34.37 %
ORIG	MED	37 281 703	6 / 18	19 414 100	47.93 %
FIX	ORIG	37 862 826	9 / 18	27 834 556	26.49 %
MED	ORIG	37 398 748	9 / 18	23 196 127	37.98 %

cycle offset strategy	best objective	better objectives	dual bound	gap
ALL	37 507 260	9 / 18	30 458 434	18.80 %
PARTIAL	37 499 535	9 / 18	30 560 248	18.51 %

Table 5: Results of Round 1 for R4L4

L	U	best objective	better objectives
FIX	MED	36 784 153	18 / 18
MED	MED	36 772 886	12 / 18
ORIG	MED	36 757 228	13 / 18
FIX	ORIG	36 770 965	12 / 18
MED	ORIG	**36 728 402**	12 / 18

cycle offset strategy	best objective	better objectives
ALL	36 775 559	12 / 18
PARTIAL	36 753 295	13 / 18

Table 6: Results of Round 2 for R4L4

To sum up, although our restricted scenarios do not always give rise to better solutions of the original instance in Round 1, the `ConcurrentPESP` solver is able to compute timetables of very good quality that can help overcome local optima on the original instance

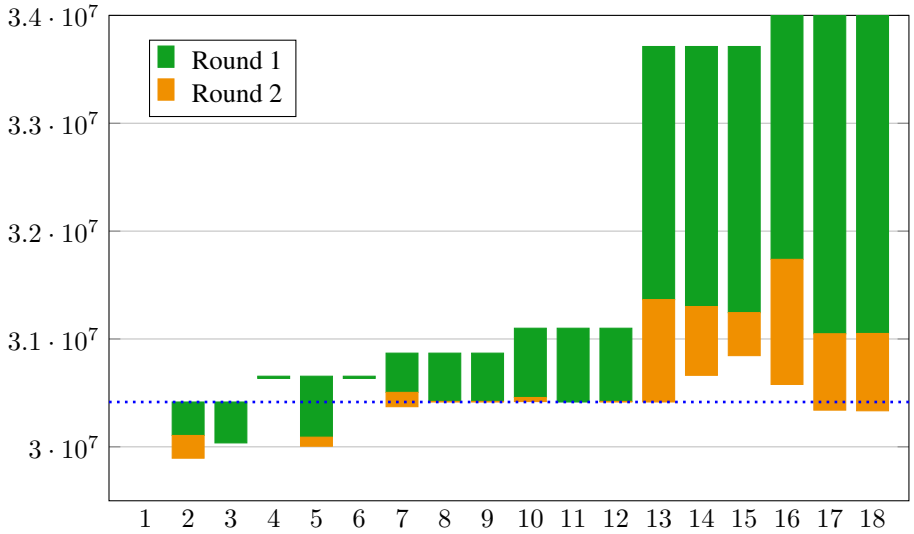

Figure 2: Objective values of the solutions obtained from R1L1(ORIG, MED). After Round 1, runs #2, #3, and #5 produced better solutions than the current PESPlib incumbent (value 30 415 672, dotted blue line). After Round 2, in total 7 better timetables have been found.

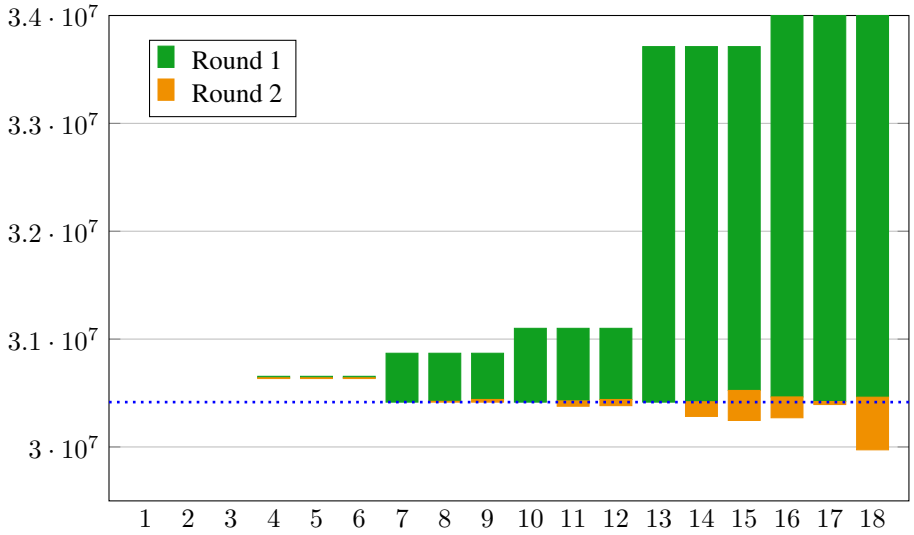

Figure 3: Objective values of the solutions obtained from MIP(R1L1, ALL). No improvement upon the current PESPlib incumbent shows up in Round 1, but 7 better timetables are found after Round 2. Interestingly, these arise from weaker initial solutions or none at all.

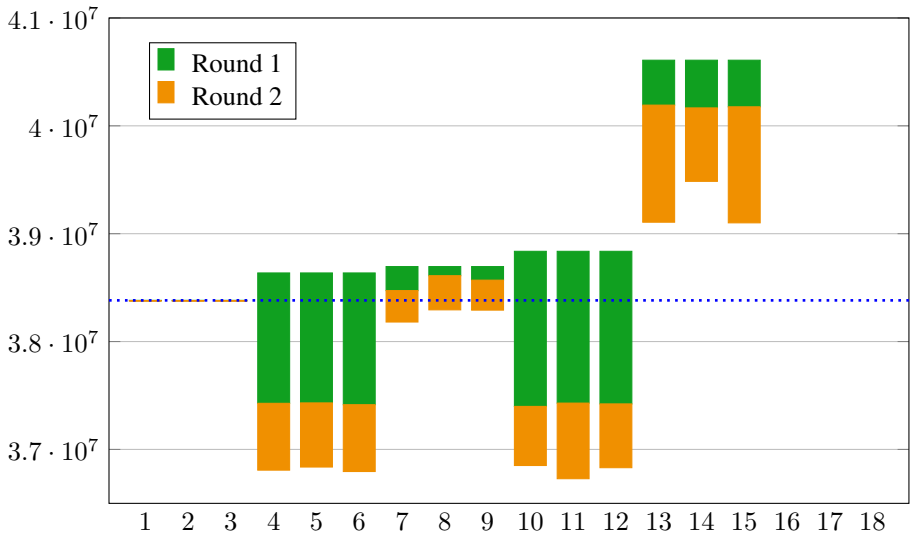

Figure 4: Objective values of the solutions obtained from R4L4(MED, ORIG). Round 1 produces 6 timetables improving upon the current PESPlib incumbent (objective value 38 381 922, dotted blue line), and 3 more are found in Round 2. Runs #16-18 have been conducted without an initial solution, and the final objective value is larger than 41 000 000.

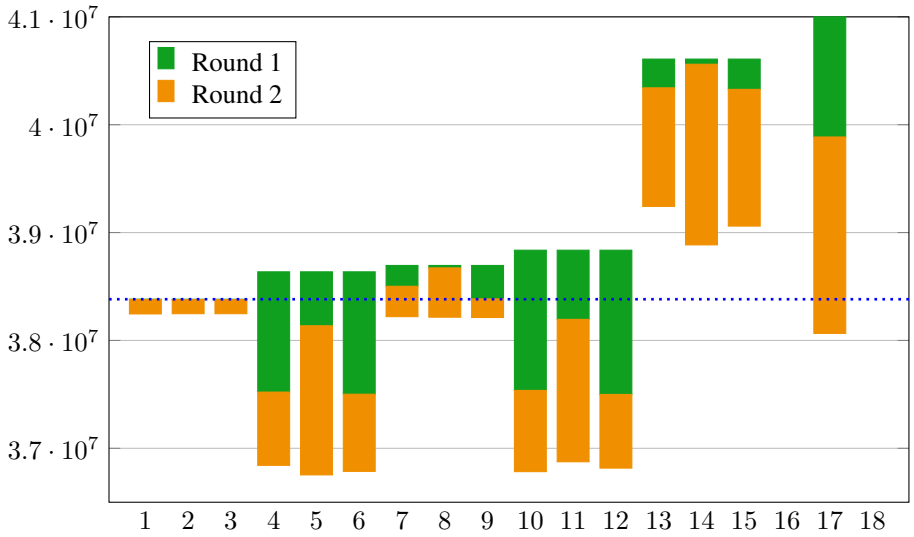

Figure 5: Objective values of the solutions obtained from MIP(R4L4, PARTIAL). After Round 2, 13 solutions have a better objective value than the current PESPlib incumbent. During runs #16 and #18 of Round 1, the solver could not find any feasible solution.

in Round 2. Restricting tensions produced lower objective values, but the difference to the cycle offset approach is only minor, so that we consider both approaches as fruitful.

5 Conclusion

We presented two strategies to merge periodic timetables in the context of the Periodic Event Scheduling Problem. Both rely on mixed integer programming, one constructing restricted PESP instances with tighter bounds on periodic tensions, and the other one defining restricted mixed-integer programs tightening the bounds of the integer cycle offset variables. The new bounds are computed using a set of initial solutions, considering the span of the activities and using minimum cycle basis techniques. On the smallest and largest PESPlib instance, this approach overcomes local optima for the classical PESP heuristics, and is therefore able to produce better periodic timetables.

We are confident that the approach will turn out to be fruitful on the other PESPlib instances, and on general PESP instances as well. Although we did not experiment with different sets S of initial solutions, the outcome was satisfactory, given the hardness of periodic timetabling problems. It would be interesting to integrate the search for these sets S into a merging framework, perhaps guided by reinforcement learning. Another direction of investigation could be to combine our two directions, i.e., restricting periodic tension and cycle offset variables simultaneously.

Bearing similarities to the crossover heuristic as it is applied for MIPs, the basic principle of our method is certainly applicable to general mixed integer programs. Note however that our computation times are several hours, whereas standard crossover for solution polishing is supposed to finish within a scale of seconds. We therefore think that the focus of such a general MIP solution merging heuristic should be on hard MIPs with general, i.e., non-binary, integer variables.

Acknowledgements

Niels Lindner was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689). We thank Enrico Bortoletto for his implementation of de Pina's minimum weight cycle basis algorithm.

References

Borndörfer, R., Lindner, N., and Roth, S. (2020). A concurrent approach to the Periodic Event Scheduling Problem. *Journal of Rail Transport Planning & Management*, 15:100175. Best Papers of RailNorrköping 2019.

Cook, W. and Seymour, P. (2003). Tour merging via branch-decomposition. *INFORMS Journal on Computing*, 15(3):233–248.

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

Goerigk, M. and Liebchen, C. (2017). An improved algorithm for the periodic timetabling

problem. In *ATMOS*, volume 59 of *OASICS*, pages 12:1–12:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Goerigk, M. and Schöbel, A. (2013). Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers & Operations Research*, 40(5):1363–1370.

Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., and Steinke, P. (2012). Solving periodic event scheduling problems with sat. In Jiang, H., Ding, W., Ali, M., and Wu, X., editors, *Advanced Research in Applied Artificial Intelligence*, pages 166–175, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hassin, R. (1996). A flow algorithm for network synchronization. *Operations Research*, 44:570–579.

Horton, J. D. (1987). A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.*, 16(2):358–366.

Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., and Zweig, K. A. (2009). Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243.

Kroon, L. G., Huisman, D., Abbink, E., Fioole, P.-J., Fischetti, M., Maroti, G., Schrijver, A., Steenbeek, A., and Ybema, R. (2009). The new Dutch timetable: The OR revolution. *INTERFACES*, 39(1):6–17.

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

Liebchen, C. (2008). The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435.

Liebchen, C. and Möhring, R. H. (2004). The modeling power of the periodic event scheduling problem: Railway timetables - and beyond. In Geraets, F., Kroon, L. G., Schöbel, A., Wagner, D., and Zaroliagis, C. D., editors, *Algorithmic Methods for Railway Optimization, International Dagstuhl Workshop, Dagstuhl Castle, Germany, June 20-25, 2004, 4th International Workshop, ATMOS 2004, Bergen, Norway, September 16-17, 2004, Revised Selected Papers*, volume 4359 of *Lecture Notes in Computer Science*, pages 3–40. Springer.

Liebchen, C. and Peeters, L. (2009). Integral cycle bases for cyclic timetabling. *Discrete Optimization*, 6(1):98–109.

Lindner, N. and Liebchen, C. (2019). New Perspectives on PESP: T-Partitions and Separators. In Cacchiani, V. and Marchetti-Spaccamela, A., editors, *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*, volume 75 of *OpenAccess Series in Informatics (OASIcs)*, pages 2:1–2:18, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Lindner, N. and Liebchen, C. (2020). Determining All Integer Vertices of the PESP Polytope by Flipping Arcs. In Huisman, D. and Zaroliagis, C. D., editors, *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*, volume 85 of *OpenAccess Series in Informatics (OASIcs)*, pages 5:1–5:18, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Lindner, N. and Reisch, J. (2020). Parameterized complexity of periodic timetabling. Technical Report 20-15, Zuse Institute Berlin.

Matos, G. P., Albino, L. M., Saldanha, R. L., and Morgado, E. M. (2020). Solving periodic timetabling problems with SAT and machine learning. *Public Transport*.

Nachtigall, K. and Opitz, J. (2008). Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations. In Fischetti, M. and Widmayer, P., editors, *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*, volume 9 of *OpenAccess Series in Informatics (OASIcs)*, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Odijk, M. A. (1994). Construction of periodic timetables, part 1: A cutting plane algorithm. Technical Report 94-61, TU Delft.

Pätzold, J. and Schöbel, A. (2016). A matching approach for periodic timetabling. In *ATMOS' 16*, volume 54 of *OASICS*, pages 1:1–1:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Roth, S. (2019). SAT heuristics for the periodic event scheduling problem. Master's Thesis, Freie Universität Berlin.

Rothberg, E. (2007). An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. *INFORMS Journal on Computing*, 19(4):534–541. Publisher: INFORMS.

Rüger, S. (1986). *Transporttechnologie städtischer öffentlicher Personenverkehr*. Transpress Verlag für Verkehrswesen, Berlin, 3rd edition.

Schiewe, P. and Schöbel, A. (2020). Periodic timetabling with integrated routing: Toward applicable approaches. *Transp. Sci.*, 54(6):1714–1731.

Serafini, P. and Ukovich, W. (1989). A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581.

Wünsch, G. (2008). *Coordination of Traffic Signals in Networks*. Ph.D. thesis, Technische Universität Berlin.