A Concurrent Approach to the Periodic Event Scheduling Problem

Ralf Borndörfer [a], Niels Lindner [a, 1], Sarah Roth [a]
[a] Zuse Institute Berlin
Takustr. 7, 14195 Berlin, Germany
[1] E-mail: lindner@zib.de, Phone: +49 30 84185374

Abstract
We introduce a concurrent solver for the periodic event scheduling problem (PESP). It combines mixed integer programming techniques, the modulo network simplex method, satisfiability approaches, and a new heuristic based on maximum cuts. Running these components in parallel speeds up the overall solution process. This enables us to significantly improve the current upper and lower bounds for all benchmark instances of the library PESPlib.

Keywords
Periodic Event Scheduling Problem, Periodic Timetabling, Mixed Integer Programming

1 Introduction

The optimization of periodic timetables is a major planning task in public transit. The standard mathematical model is the formulation as a periodic event scheduling problem (PESP, Serafini and Ukovich (1989)). In 2005, the first mathematically optimized timetable has been put into operation (Liebchen (2008)) on the Berlin subway network. However, computing optimal timetables on country-sized railway networks is notoriously hard. Therefore, the focus lies usually on feasibility rather than on minimum passenger travel time or other optimization goals (Kümmling et. al. (2015)). In particular, solving the rather large benchmark instances of the library PESPlib [1] to optimality seems currently out of reach.

The state-of-the-art methods for solving periodic timetabling problems comprise satisfiability techniques (SAT, Großmann et. al. (2012)) for feasibility questions, branch-and-cut in the framework of mixed integer programming (MIP, Liebchen (2006)), and the modulo network simplex algorithm (MNS, Nachtigall (1998)) as a local improving heuristic. In fact, the current best solutions to the PESPlib instances have been found by running a MIP solver and an MNS implementation alternatingly for 8 hours in total (Goerigk and Liebchen (2017)).

We introduce a new PESP solver based on concurrency, and integrating all three approaches. The core idea is to run MIP, MNS and a new maximum cut based heuristic in parallel. This way, the global nature of the search procedure underlying the MIP solver enables the other algorithms to escape local optima. Moreover, our solver features additional ingredients, e.g., a cutting plane separator for cycle and change-cycle inequalities.

In Section 2, we introduce PESP and two mixed integer programming formulations. The architecture and the key ingredients of our PESP solver are illustrated in Section 3. The

[1] available at `http://num.math.uni-goettingen.de/~m.goerigk/pesplib`

features of the PESPlib set and the current solution status is described in Section 4. The subsequent Section 5 contains the results of applying our solver to the PESPlib instances. We conclude this paper with a short summary in Section 6.

2 The Periodic Event Scheduling Problem

The *Periodic Event Scheduling Problem* (PESP), going back to Serafini and Ukovich (1989), is the default approach to model periodic timetabling problems. It seeks for an optimum periodic slack respecting certain bounds in a given network. We refer to Liebchen (2006) for an exhausting overview. Formally, the input for PESP is the following:

- a directed graph G with vertex set V and arc set A,
- a period time $T \in \mathbb{N}$,
- lower bounds $\ell \in \mathbb{Z}^A_{\geq 0}$,
- upper bounds $u \in \mathbb{Z}^A_{\geq 0}$, where $\ell \leq u$,
- weights $w \in \mathbb{Z}^A_{\geq 0}$.

In this paper, we will consider only integer bounds and weights. The graph G is usually called an *event-activity network*, where vertices are considered as *events*, and arcs as *activities*. Given a PESP instance (G, T, ℓ, u, w), a *periodic timetable* is an assignment of values in $[0, T)$ to the events, i.e., a vector $\pi \in [0, T)^V$. A periodic timetable defines a *periodic slack* $y \in \mathbb{R}^A_{\geq 0}$ via

$$y_{ij} := [\pi_j - \pi_i - \ell_{ij}]_T, \quad ij \in A, \tag{1}$$

where $[\cdot]_T$ denotes the modulo T operator taking values in the interval $[0, T)$. Intuitively, a periodic timetable π fixes the duration of an activity $ij \in A$ modulo T to be $[\pi_j - \pi_i]_T$, and the actual duration of ij is computed as the smallest number $\ell_{ij} + y_{ij} \in [\ell_{ij}, u_{ij}]$ satisfying $[\ell_{ij} + y_{ij}]_T = [\pi_j - \pi_i]_T$.

The PESP is now formulated as the following mixed integer program:

$$\begin{array}{lll} \text{Minimize} & w^t y & \\ \text{subject to} & y_{ij} = \pi_j - \pi_i - \ell_{ij} + p_{ij} T, & ij \in A, \\ & 0 \leq y \leq u - \ell, & \\ & 0 \leq \pi < T, & \\ & p \in \mathbb{Z}^A_{\geq 0}. & \end{array} \tag{2}$$

The integer variables p_{ij} for each activity $ij \in A$ are called *periodic offsets*. Their purpose is to model the modulo T conditions (1). Using the incidence matrix $A \in \{-1, 0, 1\}^{V \times A}$ of the network G, these constraints may as well be written as

$$y = A^t \pi - \ell + pT.$$

This is why we will call (2) the *incidence matrix mixed integer programming formulation* of PESP in the sequel. Since the incidence matrix of a directed graph is totally unimodular,

so is A^t, which implies that there is always an optimal periodic timetable π taking values in $\{0, 1, \ldots, T-1\}$. We may therefore interpret the constraint $0 \leq \pi < T$ as $0 \leq \pi \leq T-1$. A fortiori, there is always an integral optimal periodic slack y.

Another formulation can be obtained as follows: Suppose that the network G has m activities. A *cycle matrix* of G is a full row rank matrix $\Gamma \in \{-1, 0, 1\}^{\mu \times m}$ whose rows form a maximal linearly independent set of incidence vectors of oriented cycles in G, i.e., a cycle basis. If Γ represents an integral cycle basis, i.e., any maximal minor is either 0 or ± 1, then the following mixed integer program is equivalent to (2):

$$\begin{array}{lll} \text{Minimize} & & w^t y \\ \text{subject to} & & \Gamma(y + \ell) = zT \\ & & 0 \leq y \leq u - \ell, \\ & & z \in \mathbb{Z}^\mu. \end{array} \tag{3}$$

This is the *cycle matrix mixed integer programming formulation*. The number $\mu = m - n + c$ of equality constraints resp. integer variables is also called the *cyclomatic number* of G and serves as one measure of difficulty for PESP instances.

Example 1. Consider the PESP instance $I = (G, T, \ell, u, w)$ depicted in Figure 1. The timetable π indicated by the vertex labels has weighted slack $1 \cdot 5 + 5 \cdot 10 + 3 \cdot 25 = 130$. We will see later in Examples 2 and 3 that π is indeed optimal.

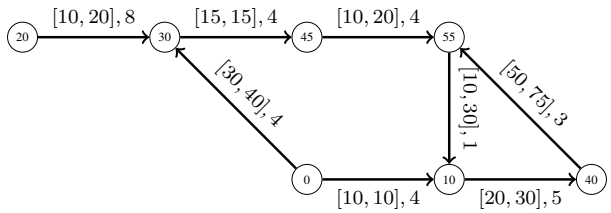


Figure 1: Example PESP instance with period time $T = 60$. The activities are labeled with $[l, u], w$. An optimal periodic timetable is given by the event labels.

3 Solver Architecture

3.1 Overview

The main idea of our PESP solver is to execute several well-performing algorithms in parallel. The solver operates in three phases, see also Fig. 2:

1. *Preprocessing phase*: Given a PESP instance, the network size is reduced by an exact preprocessing step and a subsequent heuristic preprocessing step. The exact method transforms the PESP instance into an equivalent instance – the *final problem* – with the same objective value. On the other hand, the heuristic preprocessing is allowed to slightly alter the instance and objective value, resulting in the *master problem*. The details are described in §3.2. In addition to creating the final and master problems,

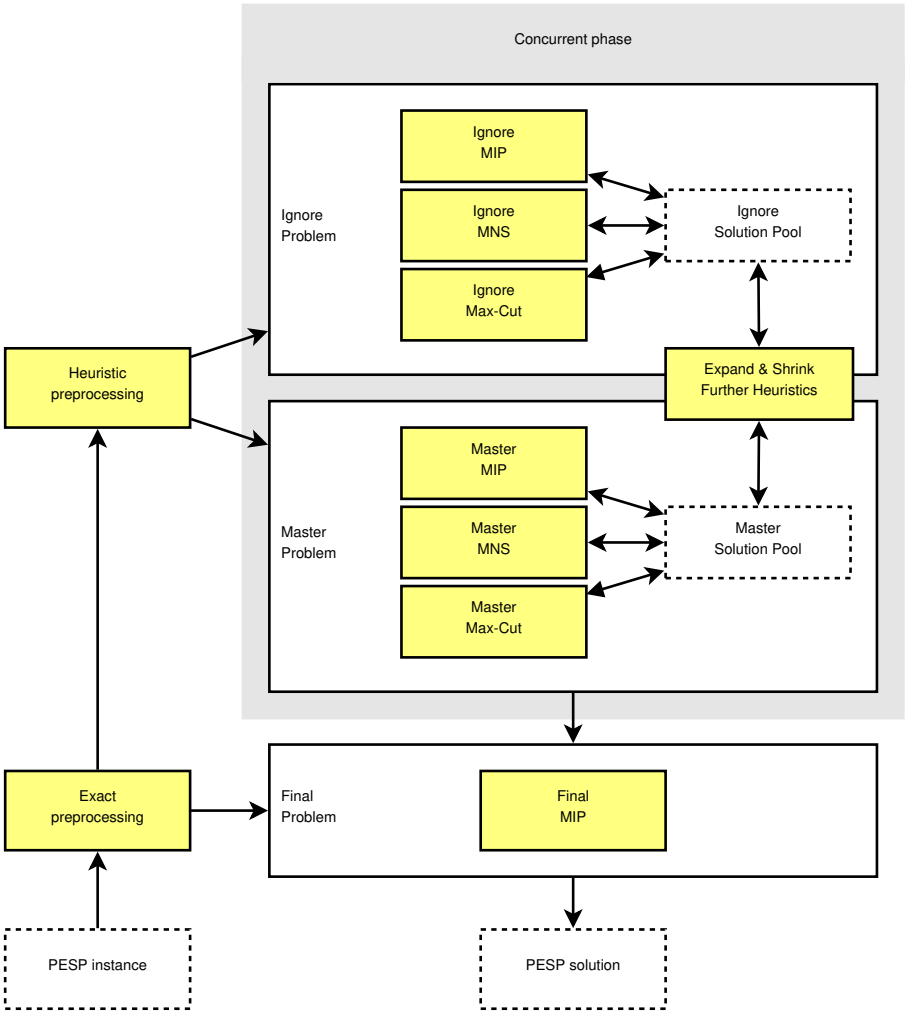


Figure 2: Architecture of our concurrent PESP solver

both preprocessing steps are also applied to a much smaller network, defining an *ignore problem*, see §3.3.

2. *Concurrent phase*: This is the main phase of the solver. Both the master and ignore problem are tackled using a MIP solver (§3.4), a modulo network simplex algorithm (§3.5), and a maximum cut heuristic (§3.6) each. These six threads run asynchronously in parallel. The incumbent solutions for each of the problems are shared among the threads by means of a common solution pool. A seventh thread transforms periodic timetables from the ignore problem's solution pool to the master problem's solution pool and vice versa, and additionally applies further heuristics (see §3.7) on both pools. While the master problem is kept for the whole concurrent phase, the ignore problem may change. The concurrent phase ends after a time limit or if the MIP solver terminates, either by detecting infeasibility or by proving optimality of the incumbent solution.

3. *Final phase*: The final problem is treated with a MIP solver, taking the best periodic timetable for the master problem as initial solution. This phase usually is aborted after a short time, as it typically does not make significant progress after the first few minutes. However, the internal heuristics of the MIP solver often detect better incumbents invisible to the master problem.

The solver is flexible in the sense that any subset of the methods in the concurrent phase can be switched off. On the other hand, the MIP solver may use several internal threads.

One advantage of concurrency is that the branch-and-cut process of the MIP does not need to wait for the other heuristics to finish and vice versa. Moreover, the communication of new solutions across all threads helps to overcome local optima.

3.2 Network Preprocessing

Our exact and heuristic preprocessing methods are based on the network size reduction strategies in Liebchen (2006) and Goerigk and Liebchen (2017). Let (G, T, ℓ, u, w) be a PESP instance. The preprocessing comprises the following steps, see also Figure 3:

1. Remove all bridges, i.e., all arcs that are not part of any oriented cycle.

2. Delete all isolated vertices.

3. Contract fixed arcs: If for an activity $a \in A$ holds $\ell_a = u_a$, then necessarily $y_a = 0$. We can hence delete a and its target j. All other arcs incident with j are replaced with arcs with the same weight from or to the source of a, adding or subtracting the lower bound ℓ_a, respectively.

4. Contract degree two vertices: If i is a vertex of degree two with an entering arc from j and a leaving arc to k, then delete i and its incident arcs, and add a new arc between j and k, adding up lower and upper bounds. The weight of the new arc becomes the minimum of the weights of the two old arcs. In exact preprocessing, this is only done if the two arcs incident to i share the same weight. However, in the case of heuristic preprocessing, also incident arcs with different weights are considered.

5. Finally, we normalize the lower and upper bounds such that both $\ell \in [0, T)$ and $u - \ell \in [0, T)$ by subtracting a suitable multiple of T. As a consequence, the periodic offsets p are then guaranteed to lie in $\{0, 1, 2\}$, reducing the size of the branch-and-bound tree the MIP solver has to search.

Both exact and heuristic preprocessing carry out all five steps. The only difference is that the heuristic preprocessing is allowed to contract more vertices of degree two. However, the preprocessing may introduce multiple arcs between two vertices.

Lemma 1. *Let $I = (G, T, \ell, u, w)$ be a PESP instance, and denote by I_{exact} and I_{heur} the PESP instances after exact and heuristic preprocessing, respectively. Then the optimal weighted slacks* OPT *satisfy*

$$\mathrm{OPT}(I) = \mathrm{OPT}(I_{exact}) \geq \mathrm{OPT}(I_{heur}).$$

Proof. As can be seen from the cycle matrix MIP formulation, there is no constraint on the periodic slack of a bridge. Hence in an optimal solution, any bridge $a \in A$ has periodic slack $y_a = 0$. Clearly, isolated vertices can be omitted. Since fixed arcs cannot have slack, they do not contribute to the objective value. Again by inspecting the cycle matrix MIP formulation, one checks that the normalization in step 5 does not affect the minimization.

It remains to check step 4. The contraction of degree two vertices does not affect the cycles of the graph, except that they contain less arcs. However, the weights may change: Let a_1, a_2 be arcs in I incident to a common degree two vertex with in-degree one and out-degree one. If their optimal periodic slacks are y_{a_1}, y_{a_2}, then the contribution to the objective value $\mathrm{OPT}(I)$ is given by $w_{a_1} y_{a_1} + w_{a_2} y_{a_2}$. The optimal solution to I can be transformed into a feasible solution to I_{heur} with the slack $y_{a_1} + y_{a_2}$ on the new arc a_{12} arising from contracting a_1 and a_2. Note that by optimality, we have $y_{a_1} + y_{a_2} < T$. However, a_{12} contributes $\min(w_{a_1}, w_{a_2})(y_{a_1} + y_{a_2})$ to $\mathrm{OPT}(I_{\text{heur}})$. This shows $\mathrm{OPT}(I) \geq \mathrm{OPT}(I_{\text{heur}})$. Observe that there is no change in objective value if $w_{a_1} = w_{a_2}$, thus $\mathrm{OPT}(I) = \mathrm{OPT}(I_{\text{exact}})$. □

Example 2. Figure 3 visualizes the preprocessing of the instance from Example 1.

In the result of the heuristic preprocessing, the bounds $[55, 75]$ and $[20, 55]$ immediately fix a duration of 55 for both arcs, and thus a duration of 5 for the third arc in reverse direction. In particular, there is only one feasible periodic slack, which is hence optimal with weighted slack 110.

Moving to the instance after exact preprocessing, there is still a single feasible periodic slack for the same reason, but now with an optimal value of 130. Tracing back the previous preprocessing steps, which only work on features not contributing to the objective value, we see that an optimal timetable for the original instance has indeed weighted slack 130.

The contraction process in step 4 can in principle be extended to all vertices of degree two. However, in this case, it is possible that $\mathrm{OPT}(I) \neq \mathrm{OPT}(I_{\text{exact}})$, see Figure 4.

As a final remark, note that the cyclomatic number of the network is preserved by all preprocessing steps. In particular, the number of equality constraints in the cycle matrix MIP formulation remains unchanged. Of course, the number of events and activities decreases and thus does the number of variables in both MIP formulations.

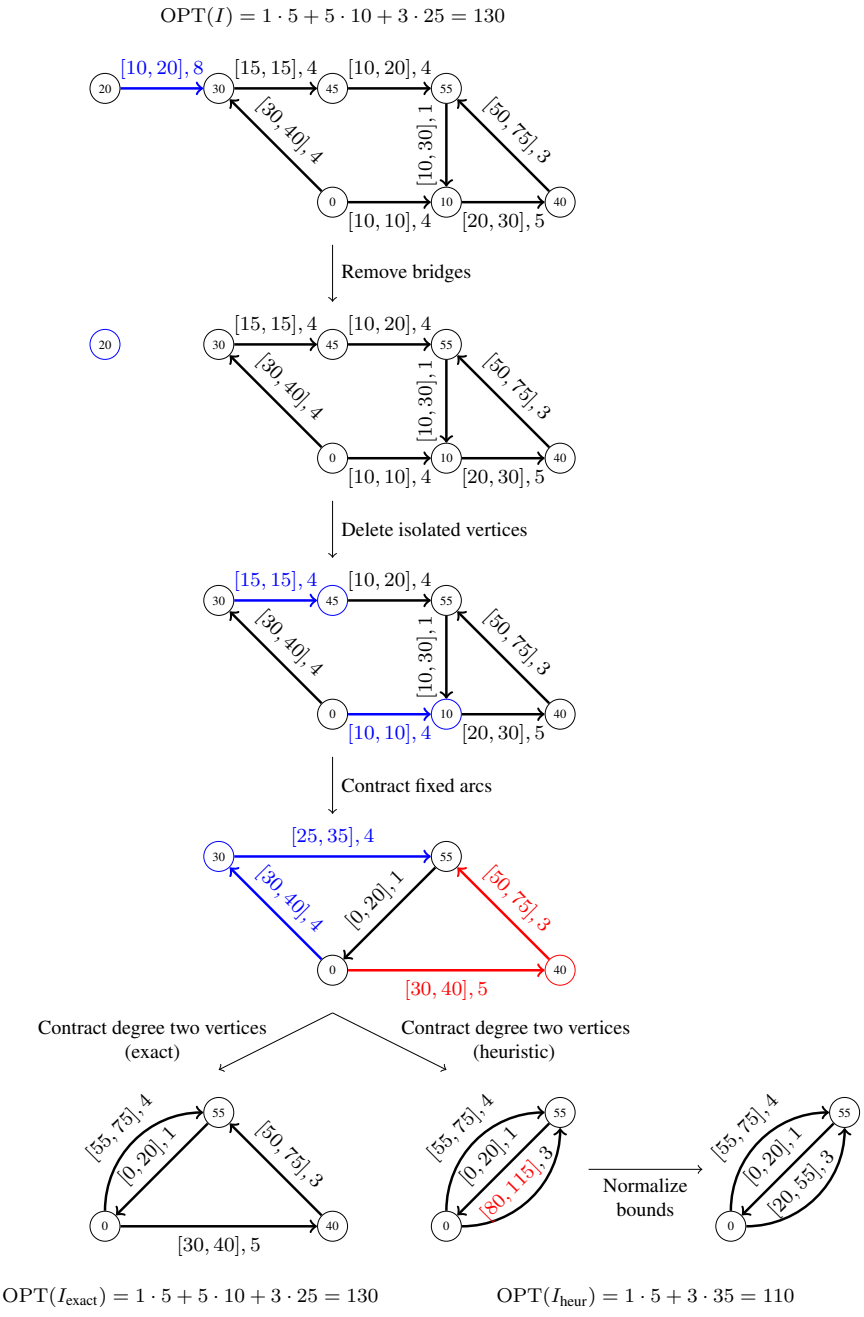
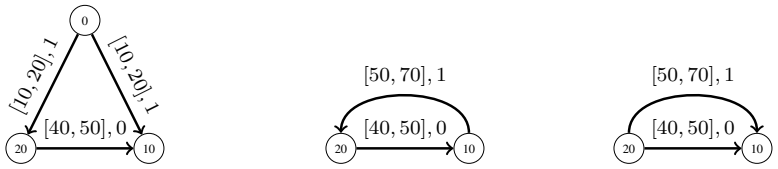


Figure 3: Exact and heuristic preprocessing



$\mathrm{OPT} = 1 \cdot 10 + 0 \cdot 10 + 1 \cdot 0 = 10$ $\mathrm{OPT} = 1 \cdot 20 + 0 \cdot 10 = 20$ $\mathrm{OPT} = 1 \cdot 0 + 0 \cdot 10 = 0$

Figure 4: Contracting arbitrary degree two vertices can lead to jumps in the optimal weighted slack.

3.3 Ignoring Free Arcs

Given a PESP instance $I = (G, T, \ell, u, w)$ and a real number $r \in [0, 1]$, we create the *ignore-r instance* I_r as follows (Goerigk and Liebchen, 2017, §3.1): Let A_{free} be the set of all *free* activities, i.e. the set of all arcs a with $u_a - \ell_a \geq T - 1$. Consider now the PESP instance arising from I by deleting the arcs from A_{free} in ascending order by weight w, until a total weight of $r \cdot \sum_{a \in A_{\text{free}}} w_a$ has been removed. Applying heuristic preprocessing to this instance defines I_r. In particular, $I_0 = I_{\text{heur}}$.

The networks I_r become smaller as r increases. Clearly, restricting a periodic timetable on I yields feasible timetable on I_r.

Lemma 2. *For any $r \in [0, 1]$ holds $\mathrm{OPT}(I_r) \leq \mathrm{OPT}(I)$.*

Proof. Any optimal solution to I is feasible for I_r, and removing arcs and heuristic preprocessing cannot increase the objective value by Lemma 1. □

Conversely, any timetable on I_r extends to a timetable on I, because the free activities are precisely the ones without any condition on their periodic slack.

Lemma 3. *Fix $r \in [0, 1]$ and let $A_{free,r}$ denote the free arcs of A removed when constructing I_r. Then*

$$\mathrm{OPT}(I) \leq \mathrm{OPT}(I_r) + \sum_{a \in A_{free,r}} w_a(T - 1) + E,$$

where E is an upper bound on the error in objective value that occurs during heuristic preprocessing of I_r.

Proof. Any optimal solution to I_r can be extended to the non-preprocessed network, causing an increase of at most E in objective value. Extending the timetable further to I adds at most $w_a(T - 1)$ for every activity a removed for constructing I_r. □

Since small terms $\sum_{A_{\text{free},r}} w_a(T - 1)$ can only be achieved with small values of r, good bounds are hard to obtain from the previous lemma. In practice, it often seems that $\mathrm{OPT}(I) \approx \mathrm{OPT}(I_r) + \sum_{A_{\text{free},r}} w_a(T - 1)/2$.

The *ignore problems* for our PESP solver stem from the ignore-r instances for different choices of $r \in [0, 1]$. The solver usually starts with a high r and decreases it after a certain amount of time. This way, the ignore problems become harder, but closer to the master problem.

3.4 Mixed Integer Programming Features

Let $I = (G, T, \ell, u, w)$ be a PESP instance. Our solver builds a mixed integer program using one of the following formulations:

- the incidence matrix formulation (§2 (2)),
- the cycle matrix formulation (§2 (3)) w.r.t. the cycle matrix of a strictly fundamental cycle basis arising from a minimum-weight spanning tree w.r.t. w,
- the cycle matrix formulation w.r.t. the cycle matrix of a minimum-weight undirected cycle basis w.r.t. $u - \ell$, if this basis is integral.

While computing a fundamental cycle basis is easily done using e.g. Kruskal's algorithm, our minimum-weight cycle basis algorithm currently relies on the rather time-consuming greedy algorithm by Horton (1987).

The MIP formulation can further be enhanced by

- *Cycle inequalities* (Odijk (1994)): Let $\gamma \in \{-1, 0, 1\}^A$ be the incidence vector of an oriented cycle. Then γ can be decomposed as $\gamma = \gamma_+ - \gamma_-$ into its non-negative and non-positive parts, i.e., $\gamma_+, \gamma_- \in \{0, 1\}^A$. The following inequality holds for any feasible periodic offset p and any feasible periodic slack y:

$$\left\lceil \frac{\gamma_+^t \ell - \gamma_-^t u}{T} \right\rceil \leq \gamma^t p = \frac{\gamma^t (y + \ell)}{T} \leq \left\lfloor \frac{\gamma_+^t u - \gamma_-^t \ell}{T} \right\rfloor$$

  If the cycle matrix formulation is used, and γ is a row of the cycle matrix Γ corresponding to an integer variable z_γ, then $\gamma^t(y + \ell)/T = z_\gamma$ and the cycle inequality yields bounds on the variable z_γ.

- *Change-cycle inequalities* (Nachtigall (1998)): Let $\gamma = \gamma_+ - \gamma_-$ be the incidence vector of an oriented cycle as above. Then the inequality

$$(T - \alpha)\, \gamma_+^t y + \alpha\, \gamma_-^t y \geq \alpha\, (T - \alpha), \qquad \text{where } \alpha = [-\gamma^t \ell]_T.$$

  holds for any feasible periodic slack y. Since the LP relaxations of both MIP formulations have 0 as their optimal values, the change-cycle inequalities are useful to provide a non-trivial lower bound for the slack variables y in the LP relaxation.

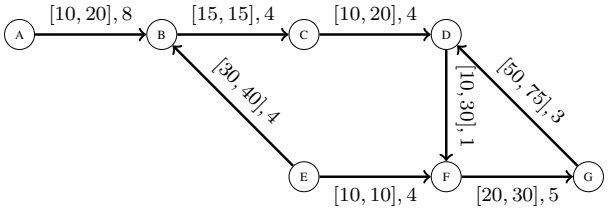


Figure 5: The PESP instance from Example 1 with named events, $T = 60$.

Example 3. We illustrate the helpfulness of the cycle inequalities on the instance of Example 1, see also Figure 5. An integral cycle basis consists of the two oriented cycles BCDFEB and DFGD, leading to the following cycle-based mixed integer program:

$$\begin{aligned}
\text{Minimize} \quad & 8y_{AB} + 4y_{BC} + 4y_{CD} + y_{DF} + 4y_{EB} + 4y_{EF} + 5y_{FG} + 3y_{GD} \\
\text{subject to} \quad & y_{BC} + y_{CD} + y_{DF} - y_{EF} + y_{EB} + 55 = 60z_1, \\
& y_{DF} + y_{FG} + y_{GD} + 80 = 60z_2, \\
& y_{BC} = y_{EF} = 0, \\
& y_{AB}, y_{CD}, y_{EB}, y_{FG} \in [0, 10], \quad y_{DF} \in [0, 20], \quad y_{GD} \in [0, 25], \\
& z_1, z_2 \in \mathbb{Z}.
\end{aligned}$$

The cycle inequalities for the two cycles read

$$\begin{aligned}
1 = \left\lceil \frac{15 + 10 + 10 - 10 + 30}{60} \right\rceil \quad &\leq z_1 \leq \left\lfloor \frac{15 + 20 + 30 - 10 + 40}{60} \right\rfloor \quad = 1 \\
2 = \left\lceil \frac{10 + 20 + 50}{60} \right\rceil \quad &\leq z_2 \leq \left\lfloor \frac{30 + 30 + 75}{60} \right\rfloor \quad = 2,
\end{aligned}$$

so that the above MIP simplifies to

$$\begin{aligned}
\text{Minimize} \quad & 8y_{AB} + 4y_{CD} + y_{DF} + 4y_{EB} + 5y_{FG} + 3y_{GD} \\
\text{subject to} \quad & y_{CD} + y_{DF} + y_{EB} = 5, \\
& y_{DF} + y_{FG} + y_{GD} = 40, \\
& y_{AB}, y_{CD}, y_{EB}, y_{FG} \in [0, 10], \quad y_{DF} \in [0, 20], \quad y_{GD} \in [0, 25].
\end{aligned}$$

The optimal solution is now to put as much slack as possible on the cheapest arc DF, i.e., to set $y_{DF} = 5$. Then $y_{FG} = 10$ and $y_{GD} = 25$, and the other slacks are 0. Consequently, the optimal solution has weighted slack $5 + 5 \cdot 10 + 3 \cdot 25 = 130$.

As a final ingredient to the MIP, we implemented a cutting plane separator. Since finding the maximally violating (change-)cycle cut is NP-hard and the best known algorithms are pseudo-polynomial dynamic programs (Borndörfer et. al. (2018)), we instead use a heuristic separator. Starting from a fractional solution to the LP relaxation, the separator adds violated (change-)cycle inequalities by inspecting the fundamental cycles of a minimum-slack spanning tree.

Currently, we interface the MIP solvers CPLEX[2] and SCIP (Gleixner et. al. (2018)).

3.5 Modulo Network Simplex

The modulo network simplex method (MNS, Nachtigall and Opitz (2008)) is an improving heuristic based on the idea that there is always an optimal PESP solution associated to a *spanning tree structure*. Formally, let $I = (G, T, \ell, u, w)$ be a PESP instance. Then there is an optimal periodic slack y^* and a spanning tree F of G such that for all arcs a in F holds either $y^*_a = 0$ or $y^*_a = u_a - \ell_a$. More precisely, the spanning tree structures correspond one-to-one to the vertices of the convex hull of

$$\{(y, z) \in \mathbb{R}^A_{\geq 0} \times \mathbb{Z}^\mu \mid \Gamma(y + \ell) = Tz, 0 \leq y \leq u - \ell\},$$

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

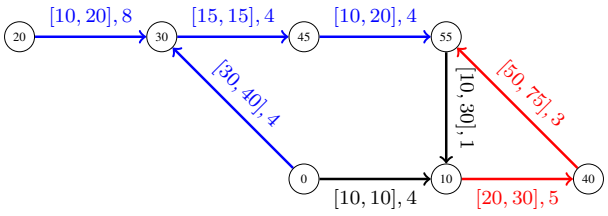


Figure 6: The PESP instance from Example 1 with an optimal spanning tree structure: Blue arcs have slack 0, red arcs have slack $u - \ell$.

i.e., the polytope associated to the cycle matrix MIP formulation (Nachtigall and Opitz, 2008, Theorem 2.1). Figure 6 illustrates an optimal spanning tree structure in our running example. Given a spanning tree structure, one can look for a better spanning tree structure by adding a co-tree arc a and deleting a tree arc along the fundamental cycle associated to a. This search can be performed with a simplex-style tableau, however, the change in objective value needs to be recomputed rather costly. After a few iterations, the MNS usually becomes stuck in a local optimum.

Our MNS implementation features the following:

1. *Initialization*: Given a feasible PESP solution, we fix the corresponding cycle offset variables z and solve the cycle matrix MIP formulation for the remaining – continuous – variables y. A strictly fundamental cycle basis computed from a minimum-weight spanning tree w.r.t. w produces the required cycle matrix. Since all integer variables are fixed, this is now a linear program, and any optimal vertex of the corresponding polytope yields a spanning tree structure.

2. *Inner loop*: Try to improve the current tree structure by exchanging tree arcs with co-tree arcs. The usual strategy is to apply steepest descent. However, the running time is drastically improved by a quality-first rule, i.e., the search for an improving move is already stopped as soon as a sufficient improvement has been achieved.

3. *Single-node cuts* (Nachtigall and Opitz (2008)): If the inner loop reaches a local optimum, then try to adjust the current timetable by modifying a single vertex.

4. *Multi-node cuts*: If inner loop and single-node cuts do not improve the timetable, then try to shift a bigger set of vertices in a random and greedy way, see Goerigk and Schöbel (2013).

5. *Restart*: If inner loop, single- and multi-node cuts, and rebuilding the spanning tree structure as in the initialization process do not lead to a better timetable, then we restart the MNS with a worse solution that was not computed by one of the MNS features. This is carried out in a tabu-search style.

The algorithm is regularly updated with the current incumbent solution, which might have been found by other algorithms, e.g., the MIP solver. The MNS turns out to be a powerful improving heuristic in the beginning of a solving process. In the later phase, improvements by the inner loop are rare, and mostly come from multi-node cuts and restarts.

3.6 Max-Cut Heuristic

Since the modulo network simplex method gets trapped in local optima too often, we developed a deeper improving heuristic dominating the MNS inner loop, single-node and multi-node cuts. Instead of searching for multi-node cuts in a heuristic way, we turn this into an optimization problem:

Problem 1 (Maximally improving delay cut)**.** Let $I = (G, T, \ell, u, w)$ be a PESP instance. We call any pair (S, d) with $S \subseteq V$ and $d \in \{1, \dots, T-1\}$ a *delay cut*. Given a feasible periodic timetable π for I, find a delay cut (S, d) such that the periodic timetable $\pi(S, d) \in [0, T)^V$ is feasible and has minimum weighted slack, where

$$\forall v \in V: \quad \pi(S,d)_v := \begin{cases} \pi_v + d & \text{if } v \in S, \\ \pi_v & \text{if } v \notin S. \end{cases}$$

Multi-node cuts are delay cuts, and clearly single-node cuts are delay cuts with a singleton subset S. Moreover, any move in the MNS inner loop can be seen as a delay cut, as removing a spanning tree arc induces a fundamental cut. In particular, if a periodic timetable cannot be improved by a delay cut, then the timetable is locally optimal for the MNS inner loop and single-/multi-node cuts.

Lemma 4. *For fixed d, the problem of finding a maximally improving delay cut is a maximum cut problem with possibly both positive and negative weights.*

Proof. We want to minimize the minimum weighted slack of $\pi(S, d)$, i.e.,

$$\sum_{ij \in A} w_{ij} [\pi(S,d)_j - \pi(S,d)_i - \ell_{ij}]_T.$$

Of course, since a fixed feasible timetable π is given, we can instead minimize

$$\sum_{ij \in A} w_{ij} \left([\pi(S,d)_j - \pi(S,d)_i - \ell_{ij}]_T - [\pi_j - \pi_i - \ell_{ij}]_T \right).$$

The summand vanishes for arcs where the endpoints are both in S or both in $V \setminus S$. Therefore, we can minimize

$$\begin{aligned} &\sum_{ij \in \delta^+(S)} w_{ij} \left([\pi_j - \pi_i - d - \ell_{ij}]_T - [\pi_j - \pi_i - \ell_{ij}]_T \right) \\ + &\sum_{ij \in \delta^-(S)} w_{ij} \left([\pi_j - \pi_i + d - \ell_{ij}]_T - [\pi_j - \pi_i - \ell_{ij}]_T \right) \end{aligned}$$

Here, $\delta^+(S)$ and $\delta^-(S)$ denote the sets of all arcs leaving S and entering S, respectively. For fixed d and π, we therefore have a minimization problem of the form

$$\sum_{ij \in \delta^+(S)} c^+_{ij} + \sum_{ij \in \delta^-(S)} c^-_{ij},$$

for fixed c^+, c^-, and we set c^+_{ij} (resp. c^-_{ij}) to ∞ if $[\pi_j - \pi_i - d - \ell_{ij}]_T$ (resp. $[\pi_j - \pi_i + d - \ell_{ij}]_T$) is not a feasible slack for the arc ij. Since in principle c^+, c^- can take any value in $(-T, T)$, we arrive at a minimum cut problem with positive and negative costs, which is at the same time a maximum cut problem by switching signs. □

The maximum cut problem as constructed in the proof is in general not solvable in polynomial time due to the presence of arcs with positive weight. To emphasize this difficulty, we prefer the term "maximum cut" over "minimum cut".

However, solving the maximally improving delay cut problem for a fixed d turns out to be well doable by a MIP solver in practice. Our PESP solver invokes SCIP to compute a maximally improving delay cut for $d = 1, 2, \ldots, \lceil (T-1)/2 \rceil$. Note that by symmetry, a delay cut (S, d) is as good as $(V \setminus S, T - d)$, and hence there is no need to check for all values of d up to $T-1$. If this max-cut heuristic (or another concurrently running algorithm) finds a better solution, it is restarted. Although this MIP-based approach is inferior to MNS in the early stage of solving, it provides better quality solutions in later phases. Using a MIP solver under the hood also enables to prove local optimality for the cut methods of MNS.

3.7 Further Ingredients

Furthermore, our solver is able to invoke the following strategies:

- *Reflow heuristic*: We apply the MNS initialization step to every new incumbent in a solution pool, not only the ones found by the modulo network simplex algorithm. Since this only involves solving a single linear program, dual to a minimum cost flow problem (Nachtigall and Opitz, 2008, §2), this is very fast. We currently use SCIP for this task.

- *SAT initial solution*: The problem of finding a feasible periodic timetable for a given PESP instance can be formulated as a boolean satisfiability (SAT) problem. This is done using the order encoding and rectangle covering strategy from Großmann et. al. (2012). Although this produces a pseudo-polynomial number of variables and clauses, a specialized SAT solver – we use *Glucose*[3] – is able to provide a feasible truth assignment typically within a second or less. The truth assignment is then transformed back to a feasible periodic timetable. We call this procedure before starting the master and the ignore problems to quickly obtain an initial solution. This is valuable since especially on larger instances, the MIP solver has difficulties to find a feasible solution in the beginning, and MNS and the max-cut heuristic require a feasible timetable as input.

- *SAT propagator*: If the MIP solver – like SCIP or CPLEX without dynamic search – applies a classical branch-and-cut algorithm to the incidence matrix MIP formulation, then the branching decisions are on the periodic offset variables p. We regularly query the current local bounds for p at the nodes of the branch-and-bound tree, and transform the PESP feasibility problem into a SAT problem as above. If the instance is infeasible – this is usually detected after a few milliseconds by glucose – we can prune the node. Otherwise we obtain another feasible solution. Unfortunately, this delays the branch-and-cut process, and the pruning effects are rather small and do not compensate turning off the dynamic search of CPLEX.

- *MaxSAT heuristic*: With a similar approach as in the feasibility case, finding an optimal periodic timetable can be translated into a weighted partial MaxSAT problem (Großmann (2016)). Since the number of clauses and variables is rather high, even a

[3] `http://www.labri.fr/perso/lsimon/glucose`

fast MaxSAT solver, like e.g. *Open-WBO* (Martins et. al. (2014)), cannot help finding good quality solutions fast. However, selecting only a small portion of arcs for the optimization is sometimes superior to a MIP approach (Roth (2019)).

4 PESPlib Instances

The library *PESPlib* serves as a benchmark set for our solver. It currently comprises 20 periodic event scheduling instances, all of which have a period time of 60 minutes. The first 16 instances arose from the German long-distance railway network. They typically decompose into disjoint paths when removing all free arcs. The last four instances are bus timetabling instances and have a different structure, e.g., there are multiple arcs between two vertices.

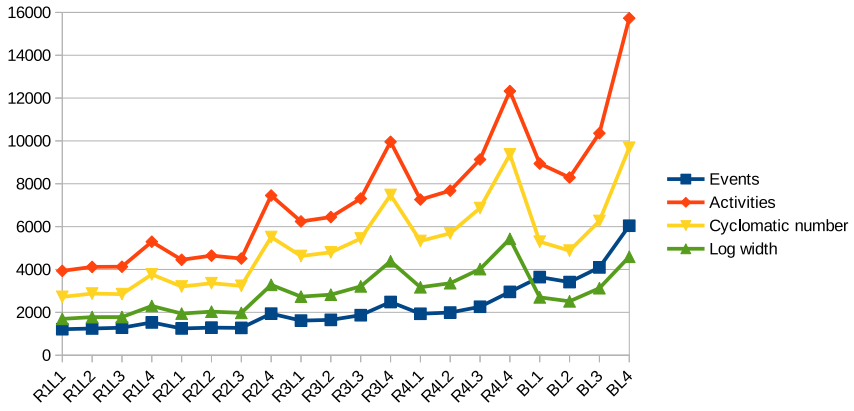


Figure 7: Sizes of PESPlib instances after heuristic preprocessing

Figure 7 compares the PESPlib instances using various measures of difficulty: number of events, number of activities, cyclomatic number and log width. Here, log width means the decadic logarithm of the number of possible values for the periodic offset vector p in the incidence matrix MIP formulation. The data refers to the networks after heuristic preprocessing. Furthermore, we transformed the instances BL1-BL4 to simple graphs, i.e., without multiple arcs. The smallest PESPlib instance R1L1 has a cyclomatic number of 2 722, and solving to proven optimality is currently out of reach. For example, a PESP instance with the rather tiny cyclomatic number 294 has found its way into the *MIPLIB2003* collection under the name *timtab2* (Liebchen and Möhring (2003)), and was solved to optimality with a pure MIP strategy within 6 432 seconds in 2016 – using the commercial MIP solver Xpress on 6 144 cores in parallel. On a single standard computer, a solving time of 22 hours with the help of special cuts is reported[4].

The effect of preprocessing is shown in Figure 8. For the first 16 instances, exact preprocessing reduces the number of events to roughly two thirds, and heuristic preprocessing

[4] http://miplib2010.zib.de/miplib2003/miplib2003/timtab2.php

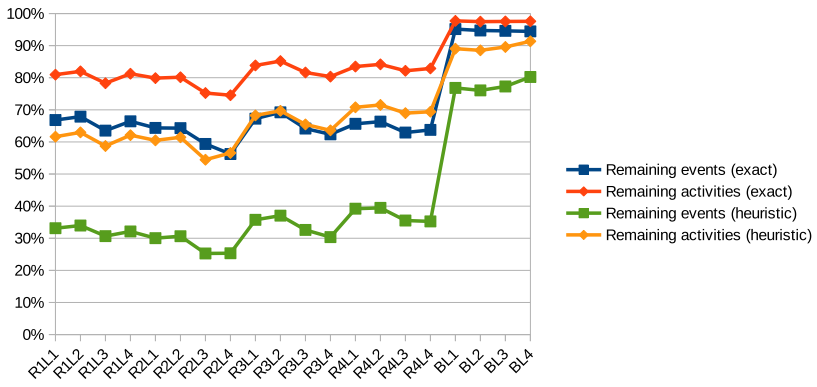


Figure 8: Size reduction by preprocessing

even to one third. The BL instances show different behavior: After heuristic preprocessing, more than 75% of all events remain.

Finally, Table 1 shows the currently best known incumbents, lower bounds, and optimality gaps on the weighted slack of all PESPlib instances. The dual bounds have received special attention only for the first instance so far.

Instance	primal	dual	gap [%]
R1L1	30 780 097	16 897 987	45.10
R1L2	31 682 263	4 975 398	84.30
R1L3	30 307 719	6 498 424	78.56
R1L4	27 326 571	6 297 850	76.95
R2L1	42 502 069	9 507 113	77.63
R2L2	41 534 563	7 768 806	81.30
R2L3	39 942 656	8 224 882	79.41
R2L4	33 063 475	5 217 025	84.22
R3L1	44 396 635	7 906 870	82.19
R3L2	46 048 483	7 432 716	83.86
R3L3	42 833 223	6 628 317	84.53
R3L4	34 694 043	5 623 632	83.79
R4L1	51 650 471	10 089 083	80.47
R4L2	49 579 843	7 975 150	83.91
R4L3	45 881 499	7 477 035	83.70
R4L4	38 836 756	5 147 195	86.75
BL1	7 387 963	1 477 565	80.00
BL2	8 143 507	1 730 247	78.75
BL3	7 826 762	1 205 501	84.60
BL4	7 359 779	1 004 303	86.35

Table 1: PESPlib incumbents as of October 24, 2018. The average optimality gap is 80.32%.

No.	Concurrent phase	Final phase	Repetitions
1	20 min	2 min	10
2	60 min	5 min	10
3	4 h	–	1
4	8 h	–	1

Table 2: Primal bound experiments

5 Computational Results

In this section, we report on the progress that our concurrent PESP solver achieved on the PESPlib instances. The computations were carried out on two machines: A 3.4 GHz Intel Xeon E3-1245 CPU and a 3.7 GHz Intel Xeon E3-1290 V2, both equipped with 32 GB RAM and allowing 8 threads running in parallel.

5.1 Primal Bound Experiments

We ran four consecutive experiments to improve the primal bounds of the PESPlib instances, see Table 2. As initial solution, the first experiment uses the timetable returned by the SAT solver (see §3.7). In particular, we construct an initial solution from scratch and do not use any input timetable from PESPlib. The first experiment spends 20 minutes in the concurrent phase and is repeated 10 times with different parameter settings for the ignore problems and the modulo network simplex quality-first strategy. The best out of this 10 solutions is taken as input for Experiment 2, which is also run with 10 different configurations. Again, the best of these solutions is taken over to Experiment 3, which is executed only once and in turn delivers its solution to Experiment 4. The experiments 3 and 4 do not enter the final phase, as the heuristic preprocessing for the master problem is switched off. In all experiments, we also do not make use of SAT methods beyond finding an initial solution.

As a MIP solver, we use CPLEX 12.8 with feasibility emphasis and dynamic search applied to the incidence matrix formulation. We add cycle inequalities before starting, but no further cuts. Experiments 1-3 use 7 threads as described in §3.1. In particular, CPLEX is run on one thread for each master and ignore problem. In Experiment 4, the ignore problem is turned off, and CPLEX uses 4 internal threads for the master problem.

The results of the experiments are summarized in Table 3. Already after Experiment 1 with 20 minutes in the concurrent phase, 10 out of 20 instances end up with a better objective value than in the PESPlib. After Experiment 2, all but the two instances R2L1 and R4L4 have been improved. In Experiments 3 and 4, the improvements become smaller, however we were able to find better incumbents for the two remaining instances as well. As the BL instances seem to have received less attention in the past, we can even improve their primal bounds by more than 10%.

Figure 9 shows the relative improvement of the objective value of the master problem for each of the heuristics ignore, MIP, MNS, max-cut, and reflow. 100% denote the total gain in objective value during the best run of the experiment. While the ignore heuristic, i.e., expanding timetables from the ignore problem, is the dominant source for new incumbents in the early stage, it has almost no effect in later phases. On the other hand, the “global” strategies such as MIP and max-cut become more important. This is why we decided to

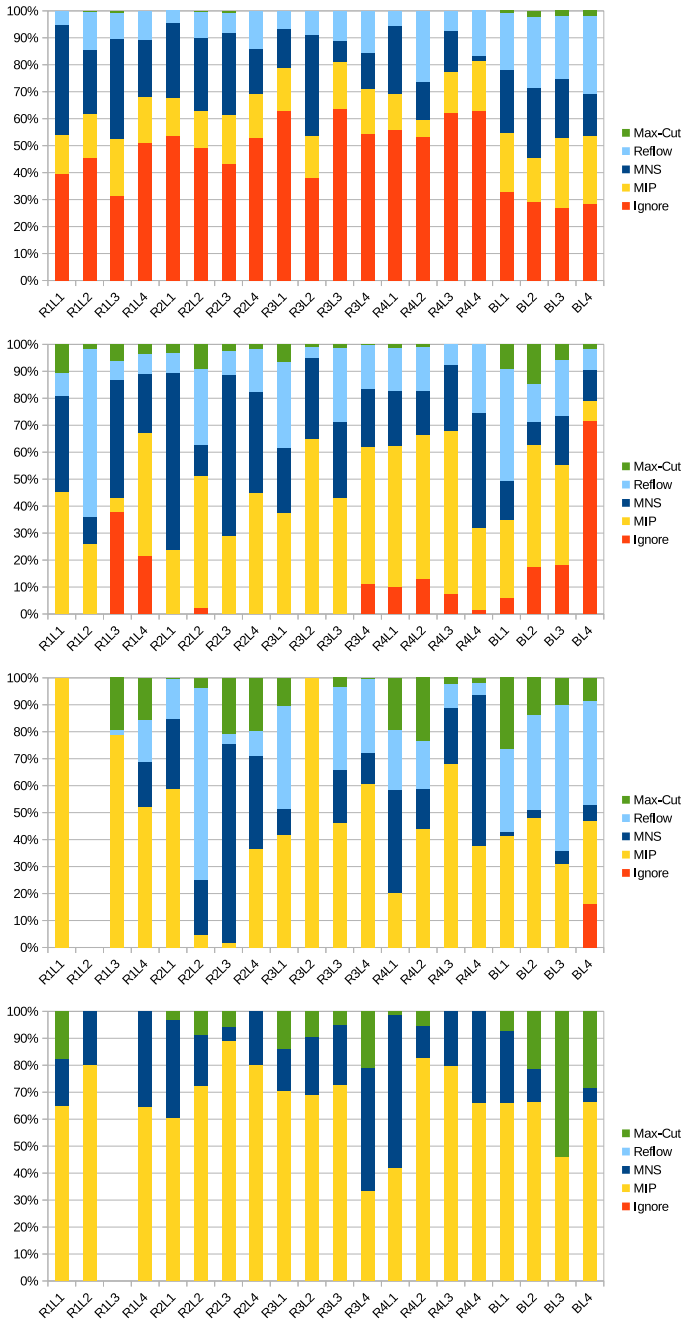


Figure 9: Relative improvement by heuristic for Experiments 1-4 from top to bottom.

Inst.	SAT start*	Exp. 1	Exp. 2	Exp. 3	Exp. 4	**
R1L1	74 234 870	30 861 021	30 501 068	30 493 800	30 463 638	1.03%
R1L2	72 731 210	30 891 284	30 516 991	30 516 991	30 507 180	3.71%
R1L3	71 682 438	30 348 596	29 335 021	29 319 593	29 319 593	3.26%
R1L4	67 395 169	27 635 070	26 738 840	26 690 573	26 516 727	2.96%
R2L1	97 230 766	42 863 646	42 598 548	42 463 738	42 422 038	0.19%
R2L2	95 898 935	42 024 414	41 149 768	40 876 575	40 642 186	2.15%
R2L3	93 800 082	39 054 513	38 924 083	38 881 659	38 558 371	3.47%
R2L4	84 605 216	33 256 602	32 707 981	32 548 415	32 483 894	1.75%
R3L1	92 939 173	44 216 552	43 521 250	43 460 397	43 271 824	2.53%
R3L2	91 336 260	45 829 180	45 442 171	45 401 718	45 220 083	1.80%
R3L3	89 741 119	42 112 858	41 103 062	41 005 379	40 849 585	4.63%
R3L4	74 142 083	34 589 170	34 018 560	33 454 773	33 335 852	3.91%
R4L1	98 276 297	50 638 727	49 970 330	49 582 677	49 426 919	4.30%
R4L2	101 135 698	50 514 805	49 379 256	49 018 380	48 764 793	1.64%
R4L3	96 629 751	46 406 365	45 656 395	45 530 113	45 493 081	0.85%
R4L4	80 446 905	40 706 349	38 884 544	38 695 188	38 381 922	1.17%
BL1	15 367 998	7 299 228	6 394 914	6 375 778	6 333 641	14.27%
BL2	16 046 736	7 378 468	6 837 447	6 819 856	6 799 331	16.51%
BL3	14 850 854	7 512 685	7 065 270	7 011 324	6 999 313	10.57%
BL4	15 618 608	7 997 783	7 330 393	6 738 582	6 562 147	10.84%

Table 3: Objective values after Experiments 1-4. Green objectives are better than in the PESPlib. All solutions to Experiment 4 are locally optimal for max-cut. *The objective value of the initial solution provided by SAT is to be interpreted on the heuristically preprocessed instance. **This is the relative improvement compared to PESPlib.

switch the ignore problem off for Experiment 4, so that 3 threads become available, which are again invested in CPLEX on the master problem.

5.2 Dual Bound Experiments

Our set-up for the primal bound experiments does not provide strong dual bounds, mostly due to the feasibility emphasis parameter setting of CPLEX. Moreover, the incidence matrix MIP formulation seems to be better for finding good primal solutions fast, but weaker concerning lower bounds. When computing a minimum-weight cycle basis and plugging in the corresponding cycle basis, we empirically observed stronger dual bounds.

For our dual bound experiment, we run CPLEX on 6 threads for 8 hours with best bound emphasis. We also invoke our heuristic cutting plane separator for cycle and change-cycle inequalities (§3.4). All other primal heuristics, e.g., MNS and max-cut, are not started. To simplify the original PESP instance I even more, we switch the master problem off and perform the computations only on the ignore problem given by the ignore-0.01 instance $I_{0.01}$ (§3.3). Since $\mathrm{OPT}(I_{0.01}) \leq \mathrm{OPT}(I)$ by Lemma 2, lower bounds on $\mathrm{OPT}(I_{0.01})$ are also valid lower bounds on $\mathrm{OPT}(I)$. Shrinking the incumbent timetable from Experiment 4 to the ignore problem provides a MIP start.

Table 4 contains the results of the dual bound experiment. We could improve all dual

bounds significantly, often by a factor of 2. As a consequence, we can reduce the average optimality gap over all instances from 80.32% to 48.36%.

Instance	Dual bound	PESPlib improvement	Optimality gap
R1L1	19 878 200	17.64%	34.75%
R1L2	19 414 800	290.22%	36.36%
R1L3	18 786 300	189.09%	35.93%
R1L4	16 822 200	167.11%	36.56%
R2L1	25 082 000	163.82%	40.88%
R2L2	24 867 400	220.09%	38.81%
R2L3	23 152 300	181.49%	39.96%
R2L4	18 941 500	263.07%	41.69%
R3L1	25 077 800	217.16%	42.05%
R3L2	25 272 600	240.02%	44.11%
R3L3	21 642 500	226.52%	47.02%
R3L4	16 479 500	193.04%	50.57%
R4L1	27 243 900	170.03%	44.88%
R4L2	26 368 200	230.63%	45.93%
R4L3	22 701 400	203.62%	50.10%
R4L4	15 840 600	207.75%	58.73%
BL1	3 668 148	148.26%	42.08%
BL2	3 943 811	127.93%	42.00%
BL3	3 571 976	196.31%	48.97%
BL4	3 131 491	211.81%	52.28%

Table 4: Dual bound experiment: Best lower bound, applied (change-)cycle cuts, relative improvement compared to the PESPlib bound, optimality gap w.r.t. the primal solution of Experiment 4. Average optimality gap over all instances: 48.36%.

6 Summary

We described a powerful framework for solving periodic event scheduling problems, providing better solutions faster, and on relatively large instances. Moreover, our approach combines many of the currently best known strategies for periodic timetabling in a single program, and it is able to compare the impact of the different methods.

Combining many state-of-the-art methods in a concurrent manner provides a significant speedup: For example, we are able to compute a new best solution to 10 out of 20 PESPlib instances in as little as 20 minutes, starting from scratch and not using any input timetable. Given the fact that many previous incumbent solutions were computed with a sequential approach using MIP and MNS for 8 hours (Goerigk and Liebchen (2017)), we achieved a speedup factor which is bigger than the number of parallel threads our solver uses.

Our cutting plane separation approach is tailor-made for improving the lower bounds on the objective values. Given that not much progress is expected in the primal bound on the PESPlib instances, we believe that the key to solve PESPlib to optimality lies in better strategies for the dual side, where even our heuristic separator is able to reduce the optimality gap significantly.

Acknowledgements
We thank C. Liebchen for sharing his PESP expertise. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689).

References

Borndörfer, R., Hoppmann, H., Karbstein, M., Lindner, N., 2018. "Separation of Cycle Inequalities in Periodic Timetabling", ZIB-Report 18-16, Zuse Institute Berlin.

Gleixner, A. et al., 2018. "The SCIP Optimization Suite 6.0", Technical Report, Optimization Online, July 2018.

Goerigk, M., Schöbel, A., 2013. "Improving the modulo simplex algorithm for large-scale periodic timetabling", *Computers & Operations Research*, vol. 40, no. 5, pp. 1363-1370.

Goerigk, M., Liebchen, C., 2017. "An improved algorithm for the periodic timetabling problem", In: *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, vol. 59, pp. 12:1-12:14.

Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., Steinke, P., 2012. "Solving Periodic Event Scheduling Problems with SAT", In: *Advanced Research in Applied Artificial Intelligence*, Springer Berlin Heidelberg, pp. 166-175.

Großmann, P., 2016. "Satisfiability and Optimization inPeriodic Traffic Flow Problems", PhD thesis, TU Dresden.

Horton, J., 1987. "A polynomial–time algorithm to find the shortest cycle basis of a graph", *SIAM J. Comput.*, vol. 16, pp. 359-366.

Kümmling, M., Großmann, P., Nachtigall, K., Opitz, J., Weiß, R., 2015. "A state-of-the-art realization of cyclic railway timetable computation", *Public Transport*, vol. 7, no. 3, pp. 281-293.

Liebchen, C., Möhring, R., 2006. "Information on MIPLIB's timetab-instances", Technical Report No. 2003/49, Technische Universität Berlin.

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

Liebchen, C., 2008. "The first optimized railway timetable in practice", *Transportation Science*, vol. 42, no. 4, pp. 420-435.

Liebchen, C., Peeters, L., 2009. "Integral cycle bases for cyclic timetabling", *Discrete Optimization*, vol. 6, no. 1, pp. 98-109.

Martins, R., Manquinho, V., Lynce, I., 2014. "Open-WBO: A Modular MaxSAT Solver", SAT 2014, pp. 438-445.

Nachtigall, K., 1998. "Periodic Network Optimization and Fixed Interval Timetables", Habilitation thesis, Universtität Hildesheim.

Nachtigall, K., Opitz, J., 2008. "Solving periodic timetable optimisation problems by modulo simplex calculations", In: *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS '08)*, vol. 9.

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

Roth, S., 2019. "SAT heuristics for periodic timetabling", Master's thesis (in preparation), Freie Universität Berlin.

Serafini, P., Ukovich, W., 1989. "A mathematical model for periodic scheduling problems", *SIAM Journal on Discrete Mathematics*, vol. 2, no. 4, pp. 550-581.