New Perspectives on PESP: T-Partitions and Separators

Niels Lindner
Zuse Institute Berlin (ZIB), Takustr. 7, 14195 Berlin, Germany
lindner@zib.de

Christian Liebchen
Technical University of Applied Sciences Wildau, Hochschulring 1, 15745 Wildau, Germany
liebchen@th-wildau.de

Abstract

In the planning process of public transportation companies, designing the timetable is among the core planning steps. In particular in the case of periodic (or cyclic) services, the Periodic Event Scheduling Problem (PESP) is well-established to compute high-quality periodic timetables.

We are considering algorithms for computing good solutions and dual bounds for the very basic PESP with no additional extra features as add-ons. The first of these algorithms generalizes several primal heuristics that have been proposed, such as single-node cuts and the modulo network simplex algorithm. We consider partitions of the graph, and identify so-called delay cuts as a structure that allows to generalize several previous heuristics. In particular, when no more improving delay cut can be found, we already know that the other heuristics could not improve either. This heuristic already had been proven to be useful in computational experiments [1], and we locate it in the more general concept of what we denote T-partitions.

With the second of these algorithms we propose to turn a strategy, that has been discussed in the past, upside-down: Instead of gluing together the network line-by-line in a bottom-up way, we develop a divide-and-conquer-like top-down approach to separate the initial problem into two easier subproblems such that the information loss along their cutset edges is as small as possible.

We are aware that there may be PESP instances that do not fit well the separator setting. Yet, on the RxLy-instances of PESPlib in our experimental computations, we come up with good primal solutions and dual bounds. In particular, on the largest instance (R4L4), this new separator approach, which applies a state-of-the-art solver as subroutine, is able to come up with better dual bounds than purely applying this state-of-the-art solver in the very same time.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Applied computing → Transportation; Mathematics of computing → Discrete optimization; Mathematics of computing → Integer programming

Keywords and phrases Periodic Event Scheduling Problem, Periodic Timetabling, Graph Partitioning, Graph Separators, Balanced Cuts

Digital Object Identifier 10.4230/OASIcs.ATMOS.2019.2

Funding *Niels Lindner*: 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).

Acknowledgements The authors want to thank Ralf Borndörfer for fruitful conversations.

1 Introduction

Traditionally, the planning process for public transportation companies is among the classical application areas of mathematical optimization. A very prominent general such success story had been established at Dutch railways [11]. At the borderline between service design and resource planning, timetabling is kind of in a central position of the entire planning process. This is one motivation why in the recent past there have been considered many "add-ons" to timetabling, e.g.,

19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019).
Editors: Valentina Cacchiani and Alberto Marchetti-Spaccamela; Article No. 2; pp. 2:1–2:18
OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- integrating decisions of line planning, sometimes even network design,
- considering the passengers' route choice as a function of the actual timetable,
- designing timetables that admit for efficient vehicle schedules and occasionally even crew schedules,
- computing delay-resistant timetables.

Nevertheless, most of these considered extensions share one limiting factor: computing efficient timetables in a core subroutine, at least.

Regarding timetables, there are several concepts around which design principles the timetable should follow, e.g., periodicity and symmetry [14]. In this paper, we are considering periodic timetables, i.e., those, in which the trips of the same line and into the same direction follow each other in a fixed time interval, which we denote the period time, or, the cycle time. In particular in Europe, these timetables are widely in use both for railways and for urban public transport.

To model periodic timetables, the Periodic Event Scheduling Problem, which had been formulated by Serafini and Ukovich [27] (see Section 2), can be considered as state-of-the-art. Notice that there are also further applications of PESP beyond periodic timetabling, such as traffic light signalling. Solution methods for PESP include (mixed) integer linear programming, constraint programming, satisfiability algorithms, as well as a couple of heuristics.

The contribution of this paper is to provide two new heuristics for computing good primal solutions for PESP instances relatively fast. On the one hand, the second heuristic does not fit for every PESP instance. On the other hand, if it fits, sometimes it can even be used to identify good dual bounds, too.

Interestingly, whereas several recent improvements to MIP performance touched on cycles within the graph model (in particular trying to improve the generally relatively weak dual bounds), both of our two heuristics deal with complementary structures, namely cutsets of a graph, in the second heuristic within the particular framework of so-called graph separators.

In Section 3, we invite the reader to think of PESP in terms of T-partitions. In particular, we introduce what we call *delay cuts* and these generalize the setting of several primal heuristics that had been considered earlier (e.g. single-node-cuts, modulo network simplex algorithm). By computing an optimal delay cut using a tailored MIP, we know that this locally optimal solution in particular is locally optimal for the other heuristics, too.

In Section 4, we propose a method to overcome some degeneracy that can sometimes be observed in a heuristic that had been dealt with in [23]. This heuristic starts, in a bottom-up manner, with optimum timetables for each line separately. Next, one combines (matches) those two clusters, between which in total find the largest weights and adjust the two separate timetables by shifting them against each other in order to synchronize the two line-clusters as good as possible. Here, sometimes it can be observed that from the moment on that one cluster becomes relatively large compared to the other clusters (still consisting of just one single directed line, in the extreme case), the heuristic degenerates simply to add – linearly – one line at a time.

This is why we are proposing to turn this procedure upside-down. At the very top level of the PESP constraint graph, we compute a separator in order to divide the instance into two essentially balanced subproblems. The two resulting PESP instances are then ideally much easier to solve to optimality. Keeping their relative structure within each of them, finally combine them to a timetable for the entire network by shifting them in a best possible way against each other. Right as in the previous approach in [23], the separator must not contain any arc that imposes a true restriction, e.g., of technical nature. Moreover, among the "free arcs", we seek for a set of arcs (and their weights) between the two subproblems that is as small as possible, in order to loose only few information.

In Section 5, we report some computational results. We start by applying the separator heuristic from Section 4. For the subproblems, we apply the concurrent solver that has been presented recently in [1], and in which the MIP-based delay cut heuristic from Section 3 is implemented among other algorithms. In these experiments, it can be observed that in the result of some separation strategies, the two separated subproblems indeed can be solved with smaller average slack than the time-equivalent benchmark solution for the original complete problem. Unfortunately, at least on the instances that we are considereing, this lead that is attained within the two subproblems turns out not to be enough to compensate the worse quality that finally appears on the arcs of the cutset that link the two subproblems when simply shifting the two pre-computed solutions of the two subproblems against each other.

2 Periodic Event Scheduling

The *Periodic Event Scheduling Problem* (PESP) is a mathematical optimization problem formulated by Serafini and Ukovich [27] that lies at the heart of periodic timetabling in public transport. The input for PESP consists of the following:

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

We will only consider integer bounds and weights in this paper. A *periodic timetable* is a vector $\pi \in \{0, 1, \dots, T-1\}^V$. Any periodic timetable defines a *periodic slack* $y \in \mathbb{Z}_{\geq 0}^A$ by

$$y_{ij} := [\pi_j - \pi_i - \ell_{ij}]_T \quad \text{for all } ij \in A,$$

where $[\cdot]_T$ denotes the modulo T operator taking values in $[0, T)$. A periodic timetable π and its associated periodic slack y are called *feasible* if $y \leq u - \ell$ holds.

In the setting of periodic timetabling for public transport, think of a period time of, say, $T = 60$ minutes. The events correspond to either the set of arrivals or departures of trips of a certain line into a particular direction. A timetable π then assigns points in time within the period time T to each of these events. Finally, the arcs measure time distances between two adjacent events, and thus model time durations for trips, stops, headways, and many more.

Given an input as above, the *Periodic Event Scheduling Problem* is now to find a feasible periodic timetable π, in an optimization version we may in addition seek for a periodic timetable minimizing the weighted slack $\sum_{ij \in A} w_{ij} y_{ij}$.

The PESP has a natural formulation as a mixed integer linear program, namely

$$\begin{aligned}
\text{Minimize} \quad & w^t y \\
\text{s.t.} \quad & y = B^t \pi - \ell + pT \\
& 0 \leq \pi \leq T - 1, \\
& 0 \leq y \leq u - \ell, \\
& p \in \mathbb{Z}^A.
\end{aligned}$$

Here, B^t denotes the transpose of the incidence matrix B of the directed graph G. Since B and hence B^t are totally unimodular [26, Example 19.2], we can w.l.o.g. relax π and y to be continuous variables.

Hence, a standard approach to solving PESP instances is to apply branch-and-cut procedures, as invoked by mixed integer programming solvers. To this end, several formulations and cutting planes have been presented [15, 17, 18, 19, 22]. Another solution strategy is to employ Boolean satisfiability methods [7, 6].

Exploiting the polyhedral structure of the problem, the modulo network simplex algorithm [20] is a rather fast local improving heuristic. Several methods for escaping local optima have been suggested [5]. We will unite these methods to a more global heuristic approach in Section 3.

Since the structure of public transportation networks is usually derived from lines, in the case when only few technical constraints have to be obeyed, a bottom-up matching approach has been introduced in [23, 12]. The idea is to cluster lines according to the importance of the transfers between them, increasing the number of lines as the matching heuristic proceeds. Doing so, it could happen that one cluster of lines is getting bigger and bigger and then, in fact, clustering only consists of a linear sequence in which the lines are added to the growing instance. In Section 4, in order to get several bigger subproblems that contain "most" of the information of the entire instance, we turn this approach upside-down: We develop a top-down divide-and-conquer strategy for PESP, i.e., we try to split the set of all lines into two parts of roughly the same size such that only a small amount of all transfers occurs between the parts. The idea is that on the intersection relatively few information is lost, whereas the practical tractability of the two subproblems improves significantly.

3 T-Partitions

In this section, we will present a view on periodic timetabling from the standpoint of cuts and partitions in graphs. Establishing a correspondence between periodic timetables and T-partitions, we translate several PESP strategies into the language of partitions. Finally, we present an improving heuristic for PESP in terms of maximum cuts, which subsumes several known local solving approaches in a single optimization problem. We identify the so-called delay cuts, as they have been already part of the computational framework presented in [1], as a useful device within the new concept of T-partitions.

3.1 Timetables and Partitions

Let (G, T, ℓ, u, w) be a PESP instance. Then any periodic timetable π naturally partitions the vertex set V of G into T sets, namely $\{i \in V \mid \pi_i = d\}$ for $d = 0, 1, \dots, T-1$.

▶ **Definition 1.** *A* T-partition *of a PESP instance with vertex set V and period time T is a T-tuple $\mathcal{V} = (V_0, V_1, \dots, V_{T-1})$ of pairwise disjoint subsets of V such that $\bigcup_{d=0}^{T-1} V_d = V$.*

Note that the members of a T-partition might be empty. Clearly, there is a one-to-one correspondence between periodic timetables and T-partitions, identifying the sets in the T-partition of V with the preimages of the periodic timetable, when interpreted as a map $V \to \{0, \dots, T-1\}$.

As periodic timetables can be thought of as maps taking values in the residue class group $(\mathbb{Z}/T\mathbb{Z}, +)$, there is a natural addition of timetables by componentwise addition modulo T. If π, π' are periodic timetables, we interpret π' as T-partition and obtain the sum as follows:

▶ **Definition 2.** *Given a periodic timetable π and a T-partition $\mathcal{V} = (V_0, \dots V_{T-1})$, define the periodic timetable $\pi^{\mathcal{V}}$ via*

$$\pi_v^{\mathcal{V}} := [\pi_v + d]_T, \quad v \in V_d, \quad d = 0, \dots, T-1.$$

We will now use T-partitions for optimizing a PESP instance. Let π^* be a timetable with minimum weighted slack. Given an initial timetable π, we can find π^* by looking for a T-partition $\mathcal{V}$ with $\pi^{\mathcal{V}} = \pi^*$. In terms of periodic slacks on the arc set A, we have:

▶ **Lemma 3.** *Let π be a periodic timetable and let $\mathcal{V}$ be a T-partition. If y and $y^{\mathcal{V}}$ are the periodic slacks associated to π and $\pi^{\mathcal{V}}$, respectively, then*

$$y^{\mathcal{V}}_{ij} = [y_{ij} - d + e]_T, \quad ij \in A \cap (V_d \times V_e), \quad d, e = 0, \dots, T-1.$$

Note that since $\mathcal{V}$ is a partition, this fixes $y^{\mathcal{V}}_{ij}$ for every arc $ij \in A$.

Proof. Plugging in the definitions,

$$y^{\mathcal{V}}_{ij} = [\pi^{\mathcal{V}}_j - \pi^{\mathcal{V}}_i + \ell_{ij}]_T = [\pi_j + e - (\pi_i + d) - \ell_{ij}]_T = [y_{ij} - d + e]_T. \qquad ◀$$

▶ **Definition 4.** *Given a periodic timetable π on a PESP instance, the* improvement *of a T-partition $\mathcal{V}$ is*

$$\iota(\pi, \mathcal{V}) := \sum_{d=0}^{T-1} \sum_{e=0}^{T-1} \sum_{ij \in A \cap (V_d \times V_e)} w_{ij} \left(y_{ij} - [y_{ij} - d + e]_T \right).$$

The Maximally Improving T-Partition *problem is to find for a given timetable π a T-partition $\mathcal{V}$ such that $\iota(\pi, \mathcal{V})$ is maximum and $y^{\mathcal{V}} \leq u - \ell$, i.e., feasible.*

▶ **Theorem 5.** *If π is a periodic timetable for a PESP instance I, then $\mathcal{V}$ solves* Maximally Improving T-Partition *for π if and only if $\pi^{\mathcal{V}}$ is an optimal solution to I.*

Proof. This follows directly from Lemma 3 and the definition of Maximally Improving T-Partition. ◀

3.2 Delay Cuts

Assuming that an initial solution is available, so far we only have transformed PESP into the equivalent Maximally Improving T-Partition problem. We will now focus on special classes of T-partitions to demonstrate the strength of this transformation. Again, we consider a PESP instance $(G = (V, A), T, \ell, u, w)$.

▶ **Definition 6.** *Let $S \subseteq V$ and $d \in \{1, \dots, T-1\}$. The T-partition $(V_0, \dots, V_{T-1})$ with*

$$V_e := \begin{cases} S & \text{if } e = d, \\ V \setminus S & \text{if } e = 0, \\ \emptyset & \text{otherwise}, \end{cases} \qquad e = 0, \dots, T-1,$$

is called a delay cut *(see [1]) with delay d and will simply be denoted by $\Delta(S, d)$. The restriction of* Maximally Improving T-Partition *to delay cuts is called the* Maximally Improving Delay Cut *problem.*

Intuitively, a delay cut $\Delta(S, d)$ delays – or shifts – all events in S by d. Delay cuts have been called *multi-node cuts* in [5], where the authors provide a way to escape from local optima produced by the modulo network simplex algorithm.

Starting with an initial timetable, an optimal timetable can be reached by decomposing a maximally improving T-partition $(V_0, V_1, \dots, V_{T-1})$ into the $T-1$ delay cuts $\Delta(V_1, 1), \dots, \Delta(V_{T-1}, T-1)$. From the perspective of T-partitions, delay cuts are hence natural building blocks. However, delay cuts themselves comprise several strategies:

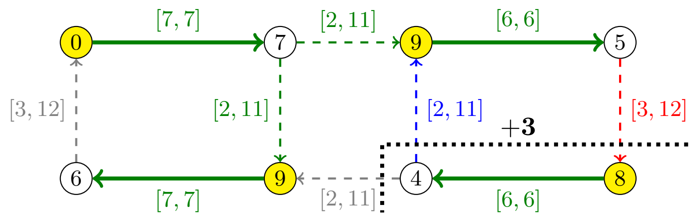

Figure 1 Fundamental delay cut: In this PESP instance with $T = 10$ and $w \equiv 1$, delaying the two vertices at the right lower corner by 3 produces a better (in fact, optimal) timetable: The overall slack is reduced from 7 to 4. This corresponds to the fundamental cut of the green spanning tree when removing the red arc. The modulo network simplex inner loop replaces the red arc with the blue arc at its lower bound.

1. *Modulo network simplex moves ("inner loop")* [20]: The key insight behind the modulo network simplex method is that there is always an optimal PESP solution coming from a spanning tree structure: There is a spanning tree (or forest if the graph is not weakly connected) such that all tree arcs have either slack 0 or $u - \ell$. Starting from such a spanning tree structure, the algorithm tries to find a better solution by exchanging a tree arc with a co-tree arc, see also [13]. The delay cut then corresponds to the fundamental cut of the tree arc, the delay depends on the co-tree arc and whether the latter is considered with slack 0 or $u - \ell$. An example is depicted in Figure 1.
2. *Single-node cuts ("outer loop")* [20], or *local improvements* [21]: These cuts are simply delay cuts $\Delta(S, d)$ with $|S| = 1$.
3. *Waiting edge cuts* [5]: If a vehicle dwells at a station where it is not terminating, then the dwell time is usually small. In particular, the difference $u - \ell$ is close to 0 and hence it seems reasonable to keep arrival and departure closely together and not to separate them by a cut. Waiting edge cuts are thus delay cuts $\Delta(S, d)$ with S consisting of the two vertices of an arc with small span $u - \ell$.

Since all these strategies rely on finding only a specific type of cut, solving Maximally Improving Delay Cut – searching the whole cut space – generalizes the above methods: If there is no improving delay cut, then also none of the approaches will be able to help. As the paper [5] only provided a randomized greedy procedure, we turn Maximally Improving Delay Cut into a genuine optimization problem.

▶ **Lemma 7.** *Let π be a periodic timetable. The improvement of a delay cut $\Delta(S, d)$ is*

$$\iota(\pi, \Delta(S, d)) = \sum_{ij \in \delta^+(S)} w_{ij}(y_{ij} - [y_{ij} - d]_T) + \sum_{ij \in \delta^-(S)} w_{ij}(y_{ij} - [y_{ij} + d]_T),$$

where $\delta^+(S)$ and $\delta^-(S)$ denote the sets of arcs leaving and entering S, respectively.

Proof. This is a simple computation from the definitions of delay cuts and the improvement of a T-partition. ◀

For a fixed delay d, we can transform Maximally Improving Delay Cut into a standard Maximum Cut problem:

1. Construct the directed graph $\overline{G}$ with vertex set $\overline{V} := V$ and arc set $\overline{A} := A \cup \{ji \mid ij \in A\}$, i.e., we add reverse copys of each arc if the reverse arc is not already present.
2. Initialize $c := 0 \in \mathbb{Z}^{\overline{A}}$.

3. For each arc $ij \in A$, set

$$c_{ij} := \begin{cases} c_{ij} + w_{ij}(y_{ij} - [y_{ij} - d]_T) & \text{if } [y_{ij} - d]_T \leq u_{ij} - \ell_{ij}, \\ -\infty & \text{otherwise.} \end{cases}$$

and

$$c_{ji} := \begin{cases} c_{ji} + w_{ij}(y_{ij} - [y_{ij} + d]_T) & \text{if } [y_{ij} + d]_T \leq u_{ij} - \ell_{ij}, \\ -\infty & \text{otherwise.} \end{cases}$$

4. Find the cut S in $\overline{G}$ such that $c(\delta^+(S))$ is maximum.

Note that since y is given and d is fixed, this is indeed a Maximum Cut problem with linear objective. As c does attain both positive and negative values, we are not able to transform our problem to a standard polynomial-time solvable Minimum Cut problem.

Although Maximum Cut is NP-hard in general [9], our problem is still easy enough to be solved on reasonably large instances within a few minutes by a MIP solver, using, e.g., the formulation presented in Appendix A. An implementation of this program invoking the solver SCIP has been included into the concurrent PESP solver presented in [1], where it proved to be successful especially when faster heuristics already got stuck in local optima.

4 Graph Separators

This section introduces a novel divide-and-conquer approach to PESP. The core idea is to split the graph into two balanced parts, on the one hand losing as little information as possible, and on the other hand obtaining subproblems which are (much) easier to solve than the entire instance, see [25] for a recent application of this concept in the context of road networks, and references therein. We can then solve the PESP restricted to each half and combine the two solutions to a solution on the original instance. In order to avoid feasibility issues, we restrict ourselves to cut the original network at *free* arcs, i.e., arcs whose slack is allowed to take arbitrary values between 0 and $T-1$. More formally, we want to find:

▶ **Definition 8.** *Let (G, T, ℓ, u, w) be a PESP instance, $G = (V, A)$. Further let $\nu : 2^V \to \mathbb{R}_{\geq 0}$ be a measure on V, and let $\alpha \geq 1$ be an imbalance parameter. A* (ν, α)-separator *is a subset $S \subseteq V$ such that*

- *$\delta(S)$ consists only of free arcs, i.e., $ij \in A$ with $u_{ij} - \ell_{ij} \geq T - 1$,*
- *$w(\delta(S))$ is minimum,*
- *$\nu(V \setminus S) \leq \nu(S) \leq \alpha \cdot \nu(V \setminus S)$.*

Here, $\delta(S)$ denotes the set of arcs in G with exactly one endpoint in S.

We will focus on the following two measures: At first, we consider balancing the number of vertices, i.e., $\nu(X) := |X|$ for $X \subseteq V$. Secondly, being a common indicator of the difficulty of a PESP instance, we will try to balance the cyclomatic number, i.e., the dimension of the cycle space of the graph, which equals $|A| - |V| + 1$ in the case of a connected graph. Of course, one could think of several more balancing criteria.

Since we are only allowed to cut through free arcs, our first step in creating a separator is to contract all non-free arcs. Note that these particular contractions are different from the commonly known PESP contractions which yield a simplified, but equivalent instance [4]. Doing so results in a multigraph, which can be resolved to a simple graph by adding up the weights of parallel arcs. The problem also permits to consider the underlying undirected graph. However, we need to keep track of the contracted vertices and the multiplicity of the arcs in order to calculate the correct measure ν, which lives on the uncontracted graph.

The structurally simplest PESP instances coming from public transit essentially contain two kinds of arcs: *Line* activities refer to driving a vehicle of a line between stations or dwelling in a station, and these activities come with only small allowed slack. *Transfer* activities are usually unconstrained in terms of slack, since one cannot expect all transfers in a network to be short, and restricting too many transfers might turn the problem infeasible. The weight on an arc is an estimate for the number of passengers using it. Recall from [16] that using PESP one is able to model a variety of other features from practice.

In this interpretation, the contraction process hence contracts all lines to single vertices. A separator then tries to divide the set of lines into two balanced parts such that the number of transferring passengers between the two parts is minimum.

We finally want to remark that finding separators is an NP-hard optimization problem in general [3]. However, it is possible to compute separators of good quality in a reasonable amount of time, and the literature is rich [2, 8, 10, 24].

4.1 Vertex-balanced Separators

By the above contraction process, finding a (ν, α)-separator balancing the number of vertices can be reduced to the following problem:

▶ **Problem 9.** Let (N, E) be an undirected graph with vertex multiplicities $m \in \mathbb{N}^N$ and edge weights $w \in \mathbb{Z}^E_{\geq 0}$. For a given imbalance $\alpha \geq 1$, find a subset $S \subseteq N$ such that
- $w(\delta(S))$ is minimum,
- $m(N \setminus S) \leq m(S) \leq \alpha \cdot m(N \setminus S)$.

▶ **Lemma 10.** *Let $n := \sum_{i \in N} m_i$. Problem 9 is solved by the mixed integer linear program*

$$\begin{aligned}
\textit{Minimize} \quad & \sum_{ij \in E} w_{ij} x_{ij} & \\
\textit{s.t.} \quad & x_{ij} \geq z_i - z_j, & ij \in E, \\
& x_{ij} \geq z_j - z_i, & ij \in E, \\
& \sum_{i \in N} m_i z_i \geq \frac{n}{2}, & \\
& \sum_{i \in N} m_i z_i \leq \frac{\alpha \cdot n}{1 + \alpha}, & \\
& x_{ij} \in [0, 1], & ij \in E, \\
& z_i \in \{0, 1\}, & i \in N.
\end{aligned}$$

Proof. See Appendix B. ◀

4.2 Cycle-balanced Separators

We will now focus on balancing the cyclomatic number μ of the parts of a PESP instance (G, T, ℓ, u, w). For a subset $X \subseteq V$, we will approximate the cyclomatic number by $\mu(X) := |A(G[X])| - |X| + 1$, where $A[G(X)]$ denotes the set of arcs of the subgraph of G induced by X. This is the exact cyclomatic number if $G[X]$ is connected, and underestimates the true quantity by the number of connected components minus one otherwise.

Since contracting arcs does not change the difference between number of arcs and vertices, we do not need to remember the number of contracted vertices for computing μ. However, collapsing parallel arcs to a simple arc decreases the cyclomatic number, so that we keep track of the multiplicity of edges.

We hence consider the following problem:

▶ **Problem 11.** Let (N, E) be an undirected graph with edge multiplicities $m \in \mathbb{N}^E$ and edge weights $w \in \mathbb{Z}^N_{\geq 0}$. For a given imbalance $\alpha \geq 1$, find a subset $S \subseteq N$ such that

- $w(\delta(S))$ is minimum,
- $\mu(N \setminus S) \leq \mu(S) \leq \alpha \cdot \mu(N \setminus S)$.

▶ **Lemma 12.** *Problem 11 can be solved by the mixed integer linear program*

$$
\begin{array}{lll}
\textit{Minimize} & \displaystyle\sum_{ij \in E} w_{ij}(1 - \ell_{ij} - r_{ij}) & \\
\textit{s.t.} & \ell_{ij} \geq z_i + z_j - 1, & ij \in E, \\
& \ell_{ij} \leq z_i, & ij \in E, \\
& \ell_{ij} \leq z_j, & ij \in E, \\
& r_{ij} \geq 1 - z_i - z_j, & ij \in E, \\
& r_{ij} \leq 1 - z_i, & ij \in E, \\
& r_{ij} \leq 1 - z_j, & ij \in E, \\
& \mu_\ell = \displaystyle\sum_{ij \in E} \ell_{ij} - \sum_{i \in N} z_i + 1, & \\
& \mu_r = \displaystyle\sum_{ij \in E} r_{ij} - \sum_{i \in N} (1 - z_i) + 1, & \\
& \mu_\ell \geq \mu_r, & \\
& \mu_\ell \leq \alpha \cdot \mu_r, & \\
& \ell_{ij} \in [0, 1], & ij \in E, \\
& r_{ij} \in [0, 1], & ij \in E, \\
& z_i \in \{0, 1\}, & i \in N.
\end{array}
$$

Proof. See Appendix B. ◀

4.3 Combining Partial Solutions

Going back to PESP instances, it is clear that restricting a feasible periodic timetable to a subgraph results in a feasible periodic timetable, and the slack cannot increase. We summarize the converse for (ν, α)-separators: Let S be a (ν, α)-separator for a PESP instance $I = (G = (V, A), T, \ell, u, w)$. Let I^ℓ, I^r, I^m be the restrictions of I to the subgraphs induced by S, $V \setminus S$ and the shores of the cut induced by S, respectively ("left", "right", "middle").

▶ **Theorem 13.** *Let S be a (ν, α)-separator, producing instances I^ℓ, I^r, I^m as above. Let π^ℓ, π^r be feasible periodic timetables for I^ℓ, I^r, respectively.*

(1) *The timetable π defined by*

$$
\pi_i := \begin{cases} \pi_i^\ell & \text{if } i \in S, \\ \pi_i^r & \text{if } i \in V \setminus S \end{cases}
$$

is feasible.

(2) *Moreover, if y^ℓ, y^r, y are the periodic slacks associated to π^ℓ, π^r, π, respectively, then*

$$
w^t y = w^t y^\ell + w^t y^m + w^t y^r,
$$

where y^m is the slack w.r.t. π of the arcs in I^m.

(3) *If* $\mathrm{opt}(J)$ *denotes the minimum weighted slack of a PESP instance* J*, then*

$$\mathrm{opt}(I^r) + \mathrm{opt}(I^m) + \mathrm{opt}(I^\ell) \leq \mathrm{opt}(I) \leq \mathrm{opt}(I^\ell) + \mathrm{opt}(I^r) + W \cdot (T-1),$$

where W *stands for the weight of the cut, i.e., sum of the weights of all arcs in* I^m*.*

Proof. Since a (ν, α)-separator cuts only through free arcs, (1) and (2) are clear. Since the optimal solution to I is feasible for the three parts I^ℓ, I^m, I^r, we obtain the left inequality. As we can combine optimal solutions to I^ℓ and I^r by (1) to a feasible solution to I, and the weighted slack increases at most by $W \cdot (T-1)$ by (2), this shows the right inequality. ◀

Therefore, these separators produce as well primal and dual bounds for PESP instances. We will demonstrate the use of separators on large-scale timetabling instances in the next section.

5 Experiments

5.1 Set-up

We use the library `PESPlib`[1] as a benchmarking set. The library contains 20 hard timetabling instances, none of which is solved to proven optimality yet. The separator strategy does not seem to be suitable for the four bus timetabling instances: When removing all free arcs, the remaining network decomposes in only 2 (BL4) or 3 (BL1-BL3) components that cannot be separated further. In other words, there are only very few possible cuts. As a consequence, only the railway instances RxLy remain, which all show a similar structure, and this is why we will focus on the easiest instance R1L1 and the hardest instance R4L4.

At first, we compute vertex-balanced separators, choosing imbalance parameters $\alpha \in \{1.05, 1.1, 1.2, 1.5\}$. To this end, we use the fast graph partitioning software `METIS` [10] to generate an initial solution and apply the MIP solver `Gurobi 8.1`[2] to the program of Lemma 10 for 20 minutes. Secondly, we determine cycle-balanced separators with the same imbalance parameters as in the vertex case. Since `METIS` cannot handle the cycle balance constraints and its solutions usually violate it, we use only `Gurobi` on the MIP of Lemma 12 for 20 minutes. Of course, for both types of separators, we contract all non-free arcs in advance, and interpret the found separator on the original network again.

Having found a separator, we solve both parts with the concurrent PESP solver from [1], which integrates mixed integer programming, modulo network simplex and the Maximally Improving Delay Cut heuristic from Section 3. This solver computed the currently best bounds for all PESPlib instances, improving 10 former primal bounds in as little as 20 minutes using 7 parallel threads. We compare these results with our separator procedure by running each part for 10 minutes with the same number of threads. In particular, the computation time on the original instance equals the sum of running times of the two parts. The reason for the small running time limit is based on the good quality of the solutions produced by the concurrent solver, and the emipirical observation that only minor improvements occur after the first 20 minutes [1]. Afterwards, we combine the timetables of both parts in an optimal way, i.e., we iterately shift the timetable of one of the parts by $0, 1, \ldots, T-1$ and choose the best combination.

1 `https://num.math.uni-goettingen.de/~m.goerigk/pesplib`
2 Gurobi Optimization LLC, `https://www.gurobi.com`

On the dual side, we compute dual bounds for each of the parts by running the concurrent solver for 10 minutes on 7 threads in pure MIP best bound mode with user cuts. We compare this with a 20-minutes run on the original instance with the same parameters.

In all PESP computations, `CPLEX 12.8`[3] serves as underlying MIP solver. The experiments were carried out on an Intel Xeon E3-1270 v6 CPU at 3.8 GHz with 32 GB RAM. For an analysis of the impact of delay cuts, we refer to [1].

5.2 Separator Statistics

Vertex-balanced separators

In every case, `Gurobi` could improve the initial vertex-balanced separator found by `METIS`. For R1L1, the vertex separators are all optimal with respect to the given imbalance, whereas optimality gaps are around 70% for R4L4. In contrast to standard minimum cuts, the smaller part sometimes consists of several connected components, which is due to the balance constraint. However, this is no issue for solving PESP. Despite having almost equal number of vertices, especially the cyclomatic number and the weights turn out to be heavily imbalanced. The smallest cuts accumulate only 19% (R1L1) resp. 24% (R4L4) of the free weight of the original instance. Table 1 resp. Table 3 contain detailed statistics about the computed separators.

Cycle-balanced separators

As no fast initial solution is available, and the program from Lemma 12 is more difficult than in the vertex case, the best optimality gaps that we can achieve after 20 minutes are 26% (R1L1) resp. 86% (R4L4). The cuts are always heavier than in the vertex case, although the difference is much smaller for the large instance R4L4. On the plus side, the solutions are much better balanced with respect to other parameters such as number of vertices, number of arcs and the free weight.

5.3 Objective Values

R1L1

For the original instance R1L1, the concurrent PESP solver was able to compute a periodic timetable with weighted slack 30 861 021 (see Table 2 for details) within 20 minutes. We typically lose a weighted slack between 10 and 18 million in the cut, so there is little space for improvement on the two parts (left and right, see rows “cut” in column “primal objective value” in Table 2). Indeed, the timetable that is computed on the full instance is superior to all combined ones. The best combined timetable has weighted slack 34 669 413, coming from a cycle-balanced separator with imbalance parameter $\alpha = 1.2$. We note that the average weighted slack on the free arcs (in particular within the cut) – which have the largest impact on the primal objective value – is significantly higher on the combined timetables than on the original. In particular, along the free arcs within the cut, average slack values of almost 50% of the period time have to be accepted, whereas in those parts for which the concurrent solver computed the timetables (original, left, right), less than 25% of the period time can be achieved as average slack.

[3] IBM ILOG CPLEX Optimization Studio, `https://www.ibm.com/analytics/cplex-optimizer`

The best dual bound computed from the sum of the two parts is 15 211 531, compared to 16 868 573. Again, the weight of the cut is the biggest hindrance, although optimality gaps are reasonably small on the parts. Due to the structure of the instance, assigning a slack of 0 to all free arcs in the cut is feasible, and we do not get any valuable lower bound from the "middle" part.

R4L4

Compared to an original primal bound of 40 706 349 after 20 minutes, we achieve 41 230 436 by a vertex-balanced separator with imbalance $\alpha = 1.2$ (see Table 4). However, all combined dual bounds (best: 11 428 968) are better than the original one (10 968 394). Thus it seems that the separator approach performs better on this larger instance. This is also due to the fact that the cuts comprise less weighted slack compared to R4L4. The good dual bound gives hope that separators might benefit to compute better lower bounds for PESP instances, which as to our experience is currently among the biggest obstacles in solving the PESPlib instances to optimality.

6 Conclusions

By considering T-partitions and introducing delay cuts for the PESP, we proposed a framework that generalizes several primal heuristics that had been known previously. In [1] the use of these cuts is already reported to contribute to the best known solutions for several instances of the PESPlib.

Regarding the separator heuristic, which can be regarded as an the entry point for a divide-and-conquer approach, so far, based on our first tuning of the computation of the separators, it is not able to come up with any better primal solutions for the instances of the PESPlib.

Nevertheless, we would not be surprised, if in the following settings the separator heuristic, too, could provide some added value:

- In contrast to the entire instance, the two resulting subproblems can be solved optimally.
- Apply the separation heuristic not only on one stage, but in a recursive, true divide-and-conquer mode.
  Yet, be aware that along the edges of each separator – although being of minimal weight – we most often observed a relatively poor quality in the final solution (almost 50% of the period time).
- We believe that diving deeper into good algorithms for graph partitioning, e.g., by using better methods or simply more running time for the mixed integer programs, could overcome the difficulty that the separators are still too heavy to provide a trade-off for improving primal and dual objectives.
- Add some kind of post-processing "around" the separator: Instead of only shifting the fixed solutions of the two subproblems as a whole against each other, just keep fixed the slack values of those edges within them which are *not* incident with the separator. Then, optimize over those timetables in which the vertices that are endpoints of an edge of the separator can be shifted relative to the subproblem that they are actually belonging to.

References

1 Ralf Borndörfer, Niels Lindner, and Sarah Roth. A Concurrent Approach to the Periodic Event Scheduling Problem. Technical Report 19-07, Zuse Institute Berlin, 2019. To appear in RailNorrköping2019 – 8th International Conference on Railway Operations Modelling and Analysis.

2 Daniel Delling, Daniel Fleischman, Andrew V. Goldberg, Ilya Razenshteyn, and Renato F. Werneck. An exact combinatorial algorithm for minimum graph bisection. *Mathematical Programming*, 153(2):417–458, November 2015.

3 M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

4 Marc Goerigk and Christian Liebchen. 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, 2017.

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

6 Peter Großmann. *Satisfiability and Optimization in Periodic Traffic Flow Problems*. PhD thesis, TU Dresden, 2016.

7 Peter Großmann, Steffen Hölldobler, Norbert Manthey, Karl Nachtigall, Jens Opitz, and Peter Steinke. Solving Periodic Event Scheduling Problems with SAT. In He Jiang, Wei Ding, Moonis Ali, and Xindong Wu, editors, *Advanced Research in Applied Artificial Intelligence*, pages 166–175, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

8 Michael Hamann and Ben Strasser. Graph Bisection with Pareto Optimization. *J. Exp. Algorithmics*, 23:1.2:1–1.2:34, February 2018.

9 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

10 George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, December 1998.

11 Leo Kroon, Dennis Huisman, Erwin Abbink, Pieter-Jan Fioole, Matteo Fischetti, Gábor Maróti, Alexander Schrijver, Adri Steenbeek, and Roelof Ybema. The new Dutch timetable: The OR revolution. *Interfaces*, 39(1):6–17, 2009.

12 Christian Liebchen. Optimierungsverfahren zur Erstellung von Taktfahrplänen. Master's thesis, Technical University Berlin, Germany, 1998. In German.

13 Christian Liebchen. A Cut-Based Heuristic to Produce Almost Feasible Periodic Railway Timetables. In Sotiris E. Nikoletseas, editor, *Experimental and Efficient Algorithms, 4th International Workshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005, Proceedings*, volume 3503 of *Lecture Notes in Computer Science*, pages 354–366. Springer, 2005. doi:10.1007/11427186_31.

14 Christian Liebchen. Optimization of passenger timetables: Are fully-integrated, regular-interval timetables really always the best? *European Rail Technical Review (RTR)*, 48(4):13–19, 2008.

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

16 Christian Liebchen and Rolf H Möhring. The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables—and Beyond. In *Algorithmic methods for railway optimization*, pages 3–40. Springer, 2007.

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

18 Christian Liebchen and Elmar Swarat. The Second Chvatal Closure Can Yield Better Railway Timetables. In Matteo Fischetti and Peter Widmayer, editors, *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*, volume 9 of *OpenAccess Series in Informatics (OASIcs)*, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

19 Karl Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. Habilitation thesis, Universität Hildesheim, 2008.

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

21 Karl Nachtigall and Stefan Voget. A genetic algorithm approach to periodic railway synchronization. *Computers & OR*, 23(5):453–463, 1996. doi:10.1016/0305-0548(95)00032-1.

22 Michiel A Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6):455–464, 1996.

23 Julius Pätzold and Anita Schöbel. A Matching Approach for Periodic Timetabling. In *ATMOS'16*, volume 54 of *OASICS*, pages 1:1–1:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

24 Peter Sanders and Christian Schulz. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, volume 7933 of *LNCS*, pages 164–175. Springer, 2013.

25 Aaron Schild and Christian Sommer. On Balanced Separators in Road Networks. In Evripidis Bampis, editor, *Experimental Algorithms - 14th International Symposium, SEA 2015, Paris, France, June 29 - July 1, 2015, Proceedings*, volume 9125 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2015. doi:10.1007/978-3-319-20086-6_22.

26 Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

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

A Maximum Cut MIP Formulation

▶ **Lemma 14.** *Let $\overline{G} = (\overline{V}, \overline{A})$ and c be as constructed in Section 3.2. A cut S in $\overline{G}$ maximizing $c(\delta^+(S))$ is computed by the following mixed integer linear program:*

$$\begin{array}{lrr} \textit{Maximize} & \sum_{ij \in \overline{A}} c_{ij} x_{ij} & \\ \textit{s.t.} & x_{ij} \leq z_i, & ij \in \overline{A}, \\ & x_{ij} \leq 1 - z_j, & ij \in \overline{A}, \\ & x_{ij} \geq z_i - z_j, & ij \in \overline{A} : c_{ij} < 0, \\ & 0 \leq x_{ij} \leq 1, & ij \in \overline{A}, \\ & z_i \in \{0,1\}, & i \in \overline{V}. \end{array}$$

Here, the variables z_i with $z_i = 1$ will define the set S.

Proof. Let (x^*, z^*) be an optimal solution to the above program. Set $S := \{i \in V \mid z_i^* = 1\}$. If $x_{ij}^* = 1$ for an arc $ij \in \overline{A}$, then $z_i^* = 1$ and $z_j^* = 0$. On the other hand, $z_i^* = 1$ and $z_j^* = 0$ imply $x_{ij}^* \geq 1$ by the third constraint for arcs with negative c_{ij} and by maximization for $c_{ij} \geq 0$. i.e., S is a cut maximizing $c(\delta^+(S)) = \sum_{ij \in \overline{A}} c_{ij} x_{ij}^*$. Conversely, a maximum cut S^* produces a feasible solution to the mixed integer program of the same cost. ◀

B Proofs of Separator MIP Formulations

Proof of Lemma 10. The constraints for the minimum cut are straightforward: A vertex i lies in S iff $z_i = 1$ and an edge ij lies in $\delta(S)$ iff $x_{ij} = 1$. We just prove the balance constraints. In order to break symmetry, we can assume $m(S) \geq m(N)/2 = n/2$, as $m(S)$ is larger than $m(N \setminus S)$. Moreover, the condition $m(S) \leq \alpha \cdot m(N \setminus S)$ directly translates to

$$\sum_{i \in N} m_i z_i \leq \alpha \sum_{i \in N} m_i (1 - z_i),$$

which is equivalent to

$$(1 + \alpha) \sum_{i \in N} m_i z_i \leq \alpha n.$$ ◀

Proof of Lemma 12. We have $z_i = 1$ iff $i \in S$, $\ell_{ij} = 1$ iff ij has both endpoints in S (ℓ for "left") and $r_{ij} = 1$ iff ij has no endpoint in S (r for "right"). The balance constraints are straightforward. ◀

C Tables

Table 1 R1L1 separator statistics: The first column contains the type of the separator, the imbalance $\alpha \in \{1.05, 1.1, 1.2, 1.5\}$ and the optimality gap. Further columns: n – number of vertices, m – number of arcs, μ – cyclomatic number, w – weight, w_{free} – weight of all free arcs, $w \cdot (u - \ell)$ – maximum possible weighted slack. Rows: original – R1L1 instance as in PESPlib, contracted – after contraction of non-free arcs, left/right – parts of the separator, cut – subgraph induced by the arcs connecting left and right.

R1L1	part	n	m	μ	w	w_{free}	$w \cdot (u - \ell)$
	original	3 664	6 385	2 722	47 172 734	2 057 406	239 600 328
	contracted	106	2 230				
vertex	left	1 876	2 927	1 052	33 725 970	1 481 768	170 793 125
1.05	right	1 788	2 058	273	12 925 650	54 524	38 061 477
0.0%	cut	1 045	1 400	516	521 114	521 114	30 745 726
vertex	left	1 918	2 990	1 073	34 412 455	1 503 621	173 692 394
1.1	right	1 746	2 004	261	12 255 217	48 723	36 109 276
0.0%	cut	1 055	1 391	499	505 062	505 062	29 798 658
vertex	left	1 996	3 205	1 210	34 847 351	1 541 460	176 996 909
1.2	right	1 668	1 870	205	11 852 913	43 476	34 727 689
0.0%	cut	1 012	1 310	459	472 470	472 470	27 875 730
vertex	left	2 198	3 609	1 412	37 061 606	1 637 281	188 155 216
1.5	right	1 466	1 598	136	9 719 139	28 136	28 317 761
0.0%	cut	969	1 178	366	391 989	391 989	23 127 351
cycle	left	1 700	2 429	730	28 474 222	1 123 356	136 712 178
1.05	right	1 964	2 663	700	18 025 146	260 684	63 159 556
33.5%	cut	949	1 293	491	673 366	673 366	39 728 594
cycle	left	1 676	2 406	731	28 180 248	1 120 386	135 658 006
1.1	right	1 988	2 718	731	18 312 779	257 313	63 839 609
34.2%	cut	967	1 261	447	679 707	679 707	40 102 713
cycle	left	1 754	2 535	782	29 076 540	1 163 077	140 590 992
1.2	right	1 910	2 562	653	17 441 343	239 478	60 373 127
36.3%	cut	979	1 288	466	654 851	654 851	38 636 209
cycle	left	1 926	2 807	882	30 359 130	1 202 889	146 190 635
1.5	right	1 738	2 327	590	16 188 820	229 733	56 547 437
26.2%	cut	955	1 251	447	624 784	624 784	36 862 256

Table 2 R1L1 objective values: Primal obj value – weighted slack of best found timetable, free % – contribution of free arcs to weighted slack, dual obj value – best lower bound, gap % – optimality gap. Rows: left, right, cut – as in Table 1, combined – optimal combination of partial timetables (primal) resp. sum of lower bounds (dual). The optimality gaps in the row combined are measured w.r.t. the best primal objective value, i.e., of the original instance.

		primal		average weighted slack			dual	
R1L1	part	obj value	free %	total	free	non-free	obj value	gap %
	original	30 861 021	87.68%	0.65	13.15	0.08	16 868 573	45.34%
vertex 1.05	left	24 894 427	88.26%	0.74	14.83	0.09	13 949 001	43.97%
	right	409 562	100.00%	0.03	7.51	0.00	358 120	12.56%
	cut	14 322 886	100.00%	27.49	27.49	–	0	–
	combined	39 626 875	92.62%	0.84	17.84	0.06	14 307 121	53.64%
vertex 1.1	left	23 376 399	87.01%	0.68	13.53	0.09	14 106 622	39.65%
	right	340 302	100.00%	0.03	6.98	0.00	295 224	13.25%
	cut	13 885 806	100.00%	27.49	27.49	–	0	–
	combined	37 602 507	91.92%	0.80	16.80	0.07	14 401 846	53.33%
vertex 1.2	left	22 842 193	86.11%	0.66	12.76	0.10	14 327 640	37.28%
	right	297 141	100.00%	0.03	6.83	0.00	256 629	13.63%
	cut	12 879 922	100.00%	27.26	27.26	–	0	–
	combined	36 019 256	91.19%	0.76	15.97	0.07	14 584 269	52.74%
vertex 1.5	left	24 857 603	86.79%	0.67	13.18	0.09	15 068 169	39.38%
	right	149 989	100.00%	0.02	5.33	0.00	143 362	4.42%
	cut	10 258 139	100.00%	26.17	26.17	–	0	–
	combined	35 265 731	90.69%	0.75	15.55	0.07	15 211 531	50.71%
cycle 1.05	left	16 382 907	85.07%	0.58	12.41	0.09	10 189 253	37.81%
	right	3 193 192	89.61%	0.18	10.98	0.02	2 608 782	18.30%
	cut	18 264 680	100.00%	27.12	27.12	–	0	–
	combined	37 840 779	92.66%	0.80	17.04	0.06	12 798 035	58.53%
cycle 1.1	left	3 288 791	91.72%	0.18	11.72	0.02	2 482 699	24.51%
	right	14 370 669	84.62%	0.51	10.85	0.08	10 110 491	29.64%
	cut	18 033 828	100.00%	26.53	26.53	–	0	–
	combined	35 693 288	93.04%	0.76	16.14	0.06	12 593 190	59.19%
cycle 1.2	left	15 029 848	86.12%	0.52	11.13	0.07	10 518 964	30.01%
	right	2 985 689	89.43%	0.17	11.15	0.02	2 341 735	21.57%
	cut	16 653 876	100.00%	25.43	25.43	–	0	–
	combined	34 669 413	93.07%	0.73	15.68	0.05	12 860 699	58.33%
cycle 1.5	left	15 523 603	84.57%	0.51	10.91	0.08	10 809 272	30.37%
	right	2 862 115	90.82%	0.18	11.31	0.02	2 218 792	22.48%
	cut	16 932 910	100.00%	27.10	27.10	–	0	–
	combined	35 318 628	92.47%	0.75	15.87	0.06	13 028 064	57.78%

Table 3 R4L4 separator statistics: See Table 1 for a legend.

R4L4	part	n	m	μ	w	w_{free}	$w \cdot (u - \ell)$
	original	8 384	17 754	9 371	65 495 305	2 219 558	297 194 946
	contracted	265	8 257				
vertex	left	4 286	8 190	3 905	35 754 908	1 013 074	151 098 512
1.05	right	4 098	6 453	2 356	29 169 656	635 743	112 422 715
70.5%	cut	1 915	3 111	1 424	570 741	570 741	33 673 719
vertex	left	4 386	8 402	4 017	36 179 480	1 032 288	153 439 283
1.1	right	3 998	6 261	2 264	28 748 028	619 473	110 255 640
72.4%	cut	1 891	3 091	1 419	567 797	567 797	33 500 023
vertex	left	4 572	8 766	4 195	37 645 797	1 076 741	159 615 340
1.2	right	3 812	5 849	2 038	27 282 163	575 472	104 106 251
75.1%	cut	1 939	3 139	1 424	567 345	567 345	33 473 355
vertex	left	5 030	9 878	4 849	41 451 476	1 252 913	180 683 746
1.5	right	3 354	4 991	1 640	23 501 054	423 870	84 487 475
73.2%	cut	1 826	2 885	1 273	542 775	542 775	32 023 725
cycle	left	4 086	7 093	3 008	33 052 401	863 684	133 516 192
1.05	right	4 298	7 204	2 907	31 792 978	705 948	125 333 120
86.0%	cut	2 097	3 457	1 596	649 926	649 926	38 345 634
cycle	left	4 796	7 941	3 146	34 722 846	824 356	138 016 435
1.1	right	3 588	6 566	2 979	30 170 267	793 010	123 649 183
84.1%	cut	1 898	3 247	1 560	602 192	602 192	35 529 328
cycle	left	4 918	8 268	3 351	36 200 384	879 816	144 508 303
1.2	right	3 466	6 265	2 800	28 684 198	729 019	116 653 986
84.8%	cut	1 863	3 221	1 574	610 723	610 723	36 032 657
cycle	left	5 098	8 891	3 794	38 255 730	951 766	154 792 427
1.5	right	3 286	5 819	2 534	26 665 459	693 676	108 529 675
87.3%	cut	1 756	3 044	1 490	574 116	574 116	33 872 844

Table 4 R4L4 objective values: See Table 2 for a legend.

		primal		average weighted slack			dual	
R4L4	part	obj value	free %	total	free	non-free	obj value	gap %
	original	40 706 349	94.69%	0.62	17.37	0.03	10 968 394	73.05%
vertex	left	15 887 937	94.18%	0.44	14.77	0.03	6 074 517	61.77%
1.05	right	10 180 187	95.79%	0.35	15.34	0.02	5 248 237	48.45%
	cut	15 936 993	100.00%	27.92	27.92	–	0	–
	combined	42 005 117	96.78%	0.64	18.32	0.02	11 322 754	72.18%
vertex	left	16 630 820	94.15%	0.46	15.17	0.03	6 025 103	63.77%
1.1	right	9 608 412	93.71%	0.33	14.53	0.02	5 141 257	46.49%
	cut	15 855 247	100.00%	27.92	27.92	–	0	–
	combined	42 094 479	96.25%	0.64	18.25	0.02	11 166 360	72.57%
vertex	left	16 923 159	94.45%	0.45	14.85	0.03	6 153 882	63.64%
1.2	right	8 133 392	89.08%	0.30	12.59	0.03	4 994 591	38.59%
	cut	16 173 885	100.00%	28.51	28.51	–	0	–
	combined	41 230 436	95.57%	0.63	17.75	0.03	11 148 473	72.61%
vertex	left	20 449 436	90.50%	0.49	14.77	0.05	6 441 593	68.50%
1.5	right	6 120 307	92.89%	0.26	13.41	0.02	3 880 625	36.59%
	cut	15 245 750	100.00%	28.09	28.09	–	0	–
	combined	41 815 493	94.31%	0.64	17.77	0.04	10 322 218	74.64%
cycle	left	12 822 538	93.21%	0.39	13.84	0.03	5 948 343	53.61%
1.05	right	11 145 363	94.12%	0.35	14.86	0.02	5 480 625	50.83%
	cut	18 328 779	100.00%	28.20	28.20	–	0	–
	combined	42 296 680	96.39%	0.65	18.37	0.02	11 428 968	71.92%
cycle	left	13 982 046	95.43%	0.40	16.19	0.02	5 736 502	58.97%
1.1	right	11 580 126	89.07%	0.38	13.01	0.04	5 460 355	52.85%
	cut	16 928 374	100.00%	28.11	28.11	–	0	–
	combined	42 490 546	95.52%	0.65	18.29	0.03	11 196 857	72.49%
cycle	left	14 648 967	94.74%	0.40	15.77	0.02	5 823 535	60.25%
1.2	right	10 313 092	86.04%	0.36	12.17	0.05	5 307 285	48.54%
	cut	17 130 851	100.00%	28.05	28.05	–	0	–
	combined	42 092 910	94.75%	0.64	17.97	0.03	11 130 820	72.66%
cycle	left	16 400 078	95.60%	0.43	16.47	0.02	6 183 490	62.30%
1.5	right	9 274 273	85.59%	0.35	11.44	0.05	5 051 562	45.53%
	cut	15 985 667	100.00%	27.84	27.84	–	0	–
	combined	41 660 018	95.06%	0.64	17.84	0.03	11 235 052	72.40%