Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

NIELS LINDNER
CHRISTIAN LIEBCHEN

New Perspectives on PESP: T-Partitions and Separators

ZIB Report 19-35 (July 2019)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

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

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

New Perspectives on PESP: T-Partitions and Separators

Niels Lindner*
Zuse Institute Berlin
lindner@zib.de

Christian Liebchen
Technical University of Applied Sciences Wildau
liebchen@th-wildau.de

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 for the very basic PESP with no additional extra features as add-ons. The first of these algorithms generalizes several primal heuristics that had been proposed in the past, 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.

The second of these algorithms turns a strategy, that had 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,

*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).

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.

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 [9]. 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.,

- 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 [13]. 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 [23] (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.

In a sense, it could be considered interesting that 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 optimum 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 [21]. There, in a bottom-up manner, one starts with optimum timetables for each line separately. Then, 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. Then, keeping their relative structure within each of them, 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 [21], the separator must not contain any arc that imposes a true restriction, e.g., of technical nature. Then, 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. There, we start by applying the separator heuristic from Section 4. For the subproblems, we apply the concurrent solver that had been presented recently in [1], and in which the MIP-based delay cut heuristic from Section 3 is implemented among other algorithms. There, 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* is a mathematical optimization problem formulated by Serafini and Ukovich [23] 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 edge set A,

- a period time $T \in \mathbb{N}$,
- lower and upper bounds $\ell, u \in \mathbb{Z}^A_{\geq 0}$ with $\ell \leq u$,
- weights $w \in \mathbb{Z}^A_{\geq 0}$.

We will only consider integer bounds and weights in this paper. A *periodic timetable* is a vector $\pi \in \{0, 1, \ldots, T-1\}^V$. Any periodic timetable defines a *periodic slack* $y \in \mathbb{Z}^A_{\geq 0}$ 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 the period time of, say, $T = 60$ minutes. The events correspond to either the set of arrivals or departures of the 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 such that the weighted slack $\sum_{ij \in A} w_{ij} y_{ij}$ that is minimum.

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 [22, 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 [12, 15, 16, 17, 20]. Another solution strategy is to employ Boolean satisfiability methods [6, 5].

Exploiting the polyhedral structure of the problem, the modulo network simplex algorithm [18] is a rather fast local improving heuristic. Several methods for escaping local optima have been suggested [4]. 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 [21, 10]. 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. There, 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.

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 $\mathcal{V} = (V_0, V_1, \dots, V_T)$ of pairwise disjoint subsets of V such that $\bigcup_{d=0}^{T-1} = V$.

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

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

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 1. *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*

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

Proof. Plugging in the definitions,

$$y_{ij}^{\mathcal{V}} = [\pi_j^{\mathcal{V}} - \pi_i^{\mathcal{V}} + \ell_{ij}]_T = [\pi_j + e - (\pi_i + d) - \ell_{ij}]_T = [y_{ij} - d + e]_T. \qquad \square$$

Definition 3. Given a periodic timetable π on a PESP instance, the *improvement* of a T-partition $\mathcal{V}$ is

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

A *maximally improving T-partition for π* is a T-partition $\mathcal{V}$ such that $\iota(\pi, \mathcal{V})$ is maximum and $y^{\mathcal{V}} \leq u - \ell$, i.e., feasible.

Theorem 1. *If π is a periodic timetable for a PESP instance I, then $\mathcal{V}$ is a 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 1 and the definition of maximally improving. $\square$

3.2. Delay Cuts

Assuming that an initial solution is available, so far we only have transformed PESP into the equivalent problem of finding a maximally improving T-partition. 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 4. Let $S \subseteq V$ and $d \in \{1, \ldots, T-1\}$. The T-partition $(V_0, \ldots, V_{T-1})$ with

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

is called a *delay cut* with delay d and will simply be denoted by (S, d).

Intuitively, a delay cut (S, d) delays – or shifts – all events in S by d. Delay cuts have been called *multi-node cuts* in [4], 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 $(V_0, V_1, \ldots, V_{T-1})$ into the $T-1$ delay cuts $(V_1, 1), \ldots, (V_{T-1}, T-1)$. From the perspective of T-partitions, delay cuts are hence natural building blocks. However, delay cuts themselves comprise several strategies:

1. *Modulo network simplex moves ("inner loop")* [18]: The key insight behind the modulo network simplex method is that there is always an optimal PESP solution coming from a spanning tree structure. I.e., there is a spanning tree (or forest if the graph is not weakly connected) such that all tree arcs have either slack 0 or

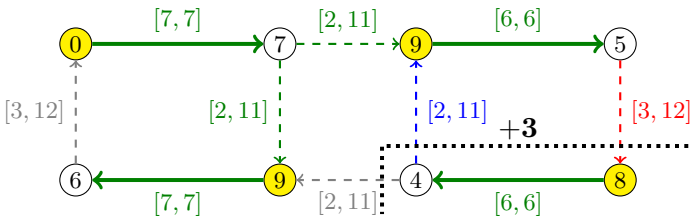

Figure 1: Fundamental delay cut: In this PESP instance with $T = 10$ and $w \equiv 1$, delaying the two vertices at the right lower corner by 3 produces a better (in fact, optimal) timetable: The overall slack is reduced from 7 to 4. This corresponds to the fundamental cut of the green spanning tree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

$u - \ell$. 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 [11]. 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")* [18], or *local improvements* [19]: These cuts are simply delay cuts (S, d) with $|S| = 1$.

3. *Waiting edge cuts* [4]: 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 (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, computing a maximally improving delay cut – searching the whole cut space – is a more global heuristic approach than all of the above methods: If there is no improving delay cut, then also none of the approaches will be able to help. As the paper [4] only provided a randomized greedy procedure, we turn the search for a maximally improving delay cut into a genuine optimization problem.

Lemma 2. *Let π be a periodic timetable. The improvement of a delay cut (S, d) is*

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

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 the maximally improving delay cut problem 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 cut problem with linear objective. Since c does attain both positive and negative values, this is more general than the standard polynomial-time solvable minimum cut problem, and thus we prefer the term *maximum cut*.

Lemma 3. *Let $\overline{G}$ be as above. For fixed d, a maximally improving delay cut is computed by the mixed integer linear program*

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

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

Proof. See Appendix A. □

Although the maximum cut problem is NP-hard in general [7], our problem is still easy enough to be solved on reasonably large instances within a few minutes by a MIP solver. An implementation of the program of Lemma 3 using the MIP 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 deals with a new 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. 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 5. Let (G, T, ℓ, u, w) be a PESP instance, $G = (V, A)$. Further let ν be some measure on the subsets of 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 [3]. 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 undirect the 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. The weight on an arc is an estimate for the number of passengers using it. Recall from [14] that using PESP one is able to model manifold further 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 [2]. However, it is possible to compute separators of good quality in a reasonable amount of time.

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:

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 4. *This problem can be solved by the mixed integer linear program*

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

where $n := \sum_{i \in N} m_i$.

Proof. See Appendix A. □

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: 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 5. *This problem can be solved by the mixed integer linear program*

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

Proof. See Appendix A. □

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 2. *With the above notation, 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 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` [8] to generate an initial solution and apply the MIP solver `Gurobi 8.1`[2] to the program

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

of Lemma 4 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 5 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 maximum cut heuristic from Section 3. This solver computed the currently best bounds for all PESPlib instances, improving 10 former primal bounds in 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. Afterwards, we combine the timetables of both parts in an optimal way, i.e., we iterate through all shifts from 0 to $T-1$ for one of the parts and choose the best combined timetable.

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 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 5 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.

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

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.

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

[3] 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.

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

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

[6] 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.

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

[8] 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.

[9] 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.

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

[11] Christian Liebchen. A cut-based heuristic to produce almost feasible periodic railway timetables. In Sotiris E. Nikoletseas, editor, *Experimental and Efficient Algorithms, 4th InternationalWorkshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005, Proceedings*, volume 3503 of *Lecture Notes in Computer Science*, pages 354–366. Springer, 2005.

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

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] Karl Nachtigall and Stefan Voget. A genetic algorithm approach to periodic railway synchronization. *Computers & OR*, 23(5):453–463, 1996.

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

[21] 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.

[22] Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency.* Springer, 2003.

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

A. Proofs of MIP Formulations

Proof of Lemma 3. 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. □

Proof of Lemma 4. 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 5. 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. □

B. Tables

R1L1	part	n	m	μ	w	w_{free}	$w \cdot (u - \ell)$
	original	3664	6385	2722	47172734	2057406	239600328
	contracted	106	2230				
vertex	left	1876	2927	1052	33725970	1481768	170793125
1.05	right	1788	2058	273	12925650	54524	38061477
0.0%	cut	1045	1400	516	521114	521114	30745726
vertex	left	1918	2990	1073	34412455	1503621	173692394
1.1	right	1746	2004	261	12255217	48723	36109276
0.0%	cut	1055	1391	499	505062	505062	29798658
vertex	left	1996	3205	1210	34847351	1541460	176996909
1.2	right	1668	1870	205	11852913	43476	34727689
0.0%	cut	1012	1310	459	472470	472470	27875730
vertex	left	2198	3609	1412	37061606	1637281	188155216
1.5	right	1466	1598	136	9719139	28136	28317761
0.0%	cut	969	1178	366	391989	391989	23127351
cycle	left	1700	2429	730	28474222	1123356	136712178
1.05	right	1964	2663	700	18025146	260684	63159556
33.5%	cut	949	1293	491	673366	673366	39728594
cycle	left	1676	2406	731	28180248	1120386	135658006
1.1	right	1988	2718	731	18312779	257313	63839609
34.2%	cut	967	1261	447	679707	679707	40102713
cycle	left	1754	2535	782	29076540	1163077	140590992
1.2	right	1910	2562	653	17441343	239478	60373127
36.3%	cut	979	1288	466	654851	654851	38636209
cycle	left	1926	2807	882	30359130	1202889	146190635
1.5	right	1738	2327	590	16188820	229733	56547437
26.2%	cut	955	1251	447	624784	624784	36862256

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	primal obj value	primal free %	average weighted slack total	average weighted slack free	average weighted slack non-free	dual obj value	dual gap %
	original	30861021	87.68%	0.65	13.15	0.08	16868573	45.34%
vertex	left	24894427	88.26%	0.74	14.83	0.09	13949001	43.97%
1.05	right	409562	100.00%	0.03	7.51	0.00	358120	12.56%
	cut	14322886	100.00%	27.49	27.49	–	0	–
	combined	39626875	92.62%	0.84	17.84	0.06	14307121	53.64%
vertex	left	23376399	87.01%	0.68	13.53	0.09	14106622	39.65%
1.1	right	340302	100.00%	0.03	6.98	0.00	295224	13.25%
	cut	13885806	100.00%	27.49	27.49	–	0	–
	combined	37602507	91.92%	0.80	16.80	0.07	14401846	53.33%
vertex	left	22842193	86.11%	0.66	12.76	0.10	14327640	37.28%
1.2	right	297141	100.00%	0.03	6.83	0.00	256629	13.63%
	cut	12879922	100.00%	27.26	27.26	–	0	–
	combined	36019256	91.19%	0.76	15.97	0.07	14584269	52.74%
vertex	left	24857603	86.79%	0.67	13.18	0.09	15068169	39.38%
1.5	right	149989	100.00%	0.02	5.33	0.00	143362	4.42%
	cut	10258139	100.00%	26.17	26.17	–	0	–
	combined	35265731	90.69%	0.75	15.55	0.07	15211531	50.71%
cycle	left	16382907	85.07%	0.58	12.41	0.09	10189253	37.81%
1.05	right	3193192	89.61%	0.18	10.98	0.02	2608782	18.30%
	cut	18264680	100.00%	27.12	27.12	–	0	–
	combined	37840779	92.66%	0.80	17.04	0.06	12798035	58.53%
cycle	left	3288791	91.72%	0.18	11.72	0.02	2482699	24.51%
1.1	right	14370669	84.62%	0.51	10.85	0.08	10110491	29.64%
	cut	18033828	100.00%	26.53	26.53	–	0	–
	combined	35693288	93.04%	0.76	16.14	0.06	12593190	59.19%
cycle	left	15029848	86.12%	0.52	11.13	0.07	10518964	30.01%
1.2	right	2985689	89.43%	0.17	11.15	0.02	2341735	21.57%
	cut	16653876	100.00%	25.43	25.43	–	0	–
	combined	34669413	93.07%	0.73	15.68	0.05	12860699	58.33%
cycle	left	15523603	84.57%	0.51	10.91	0.08	10809272	30.37%
1.5	right	2862115	90.82%	0.18	11.31	0.02	2218792	22.48%
	cut	16932910	100.00%	27.10	27.10	–	0	–
	combined	35318628	92.47%	0.75	15.87	0.06	13028064	57.78%

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.

R4L4	part	n	m	μ	w	w_{free}	$w \cdot (u - \ell)$
	original	8384	17754	9371	65495305	2219558	297194946
	contracted	265	8257				
vertex	left	4286	8190	3905	35754908	1013074	151098512
1.05	right	4098	6453	2356	29169656	635743	112422715
70.5%	cut	1915	3111	1424	570741	570741	33673719
vertex	left	4386	8402	4017	36179480	1032288	153439283
1.1	right	3998	6261	2264	28748028	619473	110255640
72.4%	cut	1891	3091	1419	567797	567797	33500023
vertex	left	4572	8766	4195	37645797	1076741	159615340
1.2	right	3812	5849	2038	27282163	575472	104106251
75.1%	cut	1939	3139	1424	567345	567345	33473355
vertex	left	5030	9878	4849	41451476	1252913	180683746
1.5	right	3354	4991	1640	23501054	423870	84487475
73.2%	cut	1826	2885	1273	542775	542775	32023725
cycle	left	4086	7093	3008	33052401	863684	133516192
1.05	right	4298	7204	2907	31792978	705948	125333120
86.0%	cut	2097	3457	1596	649926	649926	38345634
cycle	left	4796	7941	3146	34722846	824356	138016435
1.1	right	3588	6566	2979	30170267	793010	123649183
84.1%	cut	1898	3247	1560	602192	602192	35529328
cycle	left	4918	8268	3351	36200384	879816	144508303
1.2	right	3466	6265	2800	28684198	729019	116653986
84.8%	cut	1863	3221	1574	610723	610723	36032657
cycle	left	5098	8891	3794	38255730	951766	154792427
1.5	right	3286	5819	2534	26665459	693676	108529675
87.3%	cut	1756	3044	1490	574116	574116	33872844

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

		primal		average weighted slack			dual	
R4L4	part	obj value	free %	total	free	non-free	obj value	gap %
	original	40706349	94.69%	0.62	17.37	0.03	10968394	73.05%
vertex	left	15887937	94.18%	0.44	14.77	0.03	6074517	61.77%
1.05	right	10180187	95.79%	0.35	15.34	0.02	5248237	48.45%
	cut	15936993	100.00%	27.92	27.92	–	0	–
	combined	42005117	96.78%	0.64	18.32	0.02	11322754	72.18%
vertex	left	16630820	94.15%	0.46	15.17	0.03	6025103	63.77%
1.1	right	9608412	93.71%	0.33	14.53	0.02	5141257	46.49%
	cut	15855247	100.00%	27.92	27.92	–	0	–
	combined	42094479	96.25%	0.64	18.25	0.02	11166360	72.57%
vertex	left	16923159	94.45%	0.45	14.85	0.03	6153882	63.64%
1.2	right	8133392	89.08%	0.30	12.59	0.03	4994591	38.59%
	cut	16173885	100.00%	28.51	28.51	–	0	–
	combined	41230436	95.57%	0.63	17.75	0.03	11148473	72.61%
vertex	left	20449436	90.50%	0.49	14.77	0.05	6441593	68.50%
1.5	right	6120307	92.89%	0.26	13.41	0.02	3880625	36.59%
	cut	15245750	100.00%	28.09	28.09	–	0	–
	combined	41815493	94.31%	0.64	17.77	0.04	10322218	74.64%
cycle	left	12822538	93.21%	0.39	13.84	0.03	5948343	53.61%
1.05	right	11145363	94.12%	0.35	14.86	0.02	5480625	50.83%
	cut	18328779	100.00%	28.20	28.20	–	0	–
	combined	42296680	96.39%	0.65	18.37	0.02	11428968	71.92%
cycle	left	13982046	95.43%	0.40	16.19	0.02	5736502	58.97%
1.1	right	11580126	89.07%	0.38	13.01	0.04	5460355	52.85%
	cut	16928374	100.00%	28.11	28.11	–	0	–
	combined	42490546	95.52%	0.65	18.29	0.03	11196857	72.49%
cycle	left	14648967	94.74%	0.40	15.77	0.02	5823535	60.25%
1.2	right	10313092	86.04%	0.36	12.17	0.05	5307285	48.54%
	cut	17130851	100.00%	28.05	28.05	–	0	–
	combined	42092910	94.75%	0.64	17.97	0.03	11130820	72.66%
cycle	left	16400078	95.60%	0.43	16.47	0.02	6183490	62.30%
1.5	right	9274273	85.59%	0.35	11.44	0.05	5051562	45.53%
	cut	15985667	100.00%	27.84	27.84	–	0	–
	combined	41660018	95.06%	0.64	17.84	0.03	11235052	72.40%

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