A Geometric Approach to Integrated Periodic Timetabling and Passenger Routing

Fabian Löbel
Zuse Institute Berlin, Germany

Niels Lindner
Freie Universität Berlin, Germany

Abstract

We offer a geometric perspective on the problem of integrated periodic timetabling and passenger routing in public transport. Inside the space of periodic tensions, we single out those regions, where the same set of paths provides shortest passenger routes. This results in a polyhedral subdivision, which we combine with the known decomposition by polytropes. On each maximal region of the common refinement, the integrated problem is solvable in polynomial time. We transform these insights into a new geometry-driven primal heuristic, integrated tropical neighborhood search (ITNS). Computationally, we compare implementations of ITNS and the integrated (restricted) modulo network simplex algorithm on the TimPassLib benchmark set, and contribute better solutions in terms of total travel time for all but one of the twenty-five instances for which a proven optimal solution is not yet known.

2012 ACM Subject Classification Applied computing → Transportation; Mathematics of computing → Combinatorial optimization; Theory of computation → Network optimization

Keywords and phrases Periodic Timetabling, Passenger Routing, Polyhedral Complexes

Digital Object Identifier 10.4230/OASIcs.ATMOS.2025.10

Funding *Fabian Löbel*: 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).

1 Introduction

Public transport systems constitute the backbone of urban mobility in many areas, some of which are used by several million daily passengers. A skillful design of those systems is not only important for their regular users, but is also key to attract more passengers, which is a designated goal to improve sustainability, space consumption, and overall efficiency of mobility. Ideally, individuals will base their mode and route choice on their expected travel time. One key characteristic of efficient public transport is to bundle passengers on similar routes, that are operated with discrete frequencies. Therefore, it is unavoidable that passengers will spend some time waiting, e.g., before being able to board the next vehicle. For a public transport operator, it is thus necessary to offer a service that balances the needs of the passengers, so that the journey duration is adequate for most of the users.

For example, when creating a timetable, it is a reasonable goal to minimize the total travel time for all passengers. Unfortunately, periodic timetable optimization, which is mathematically often formulated in terms of the *Periodic Event Scheduling Problem* (PESP) [9, 21], is a very challenging NP-hard problem [14]. Even worse, the PESP model assumes that passengers always use the same route, regardless of the timetable. In practice, the opposite is true: Passengers choose their paths through the public transport network by the currently operated timetable. Consequently, it is only natural to include the route choice of passengers into the timetabling problem.

In this paper, we therefore investigate the problem of *Integrated Periodic Timetabling and Passenger Routing* (TimPass) [1, 6, 17, 18, 19, 20, 22]. The problem can be formulated as a bilinear mixed integer program and is hard to solve to optimality in practice. While a vast supply of heuristics for periodic timetabling is known [3], this is less true for the integrated problem, although there is some literature [11, 15, 16, 18]. Of course, a straightforward technique is to design iterative approaches, that compute a passenger routing, then improve the timetable for that routing, until at some point the passenger routes are changed again, and so forth. In real-world instances, not only the timetabling subproblem is difficult, but also frequent shortest path computations, that are necessary to evaluate the actual quality of a timetable, turn out to be a huge computational bottleneck. The sweet spot is hence to decide when to keep the current routing, and when to recompute passenger paths. For example, the *modulo network simplex* (MNS) algorithm [13] can be generalized to a family of heuristic algorithms for TimPass that differ in the frequency of shortest path computations [10].

Our theoretical main contribution is a geometric answer to this question. The space of solutions to the TimPass problem can be parametrized in terms of periodic tensions, i.e., the collection of durations of activities such as driving between two stations, dwelling inside a vehicle, or performing a transfer. This space admits a polyhedral decomposition by regions, such that for all tensions inside a region, the same set of paths is a shortest path for all origin-destination pairs [8]. Moreover, the tension space is known to admit a decomposition into polytropes [5]. The latter insight has led to *tropical neighborhood search* (TNS), a geometry-inspired PESP heuristic [4]. Considering the common refinement of the shortest path subdivision with the polytrope decomposition, we obtain a subdivision of the periodic tension space such that on each region, the TimPass problem becomes polynomial-time solvable (Corollary 14). When restricting to a polytrope, the computation of shortest paths for a single origin-destination pair can already be simplified (Lemma 11, Theorem 12, Corollary 13).

On the practical side, we introduce *integrated tropical neighborhood search* (ITNS), in a coarse and in a fine variant. The geometric idea is to solve TimPass on a region of the subdivision, and then to scan neighboring regions for improvements. We benchmark ITNS against integrated MNS techniques on the TimPass benchmark set [17]. As a result, we obtain better solutions for six of the instances within an hour on a regular desktop workstation, and can improve on the best known solution for all but three instances by taking advantage of parallelization on a compute cluster. Note that two of those three instances have already been solved to optimality previously.

The paper is structured as follows: We review the construction and introduce our notation of extended event-activity networks in Section 2, which allows us to define the TimPass problem. In Section 3, we recall the (restricted) integrated modulo network simplex algorithm. The geometric picture is unfolded in Section 4, leading to the integrated tropical neighborhood search algorithm presented in Section 5. Our computational results on the TimPassLib instances are evaluated in Section 6. We conclude the paper in Section 7.

2 Problem Modeling

This paper revolves around the *Integrated Periodic Timetabling and Passenger Routing Problem* (TimPass). It is based on a directed graph, which we call the *extended event-activity network*. We briefly explain the features of such networks in Section 2.1, before describing the TimPass optimization problem in Section 2.2.

2.1 Extended Event-Activity Networks

Originally proposed in [21], an *event-activity network* is a directed graph $G = (V, A)$ whose vertices are called *events* and whose arcs are called *activities*.

We are particularly concerned with timetabling in public transport. Each line or trip of a public transport network admits an alternating sequence of *departure* and *arrival events*, connected by an alternating sequence of *driving* and *dwelling* activities. Relations between different trips can be described by further activities, such as *transfer activities* that model passenger transfers, or *headway activities* that ensure a certain time distance between two events. An example network is depicted in Figure 1. The goal of timetabling is then to assign times to the events such that the resulting activity duration between adjacent events is within pre-specified bounds.

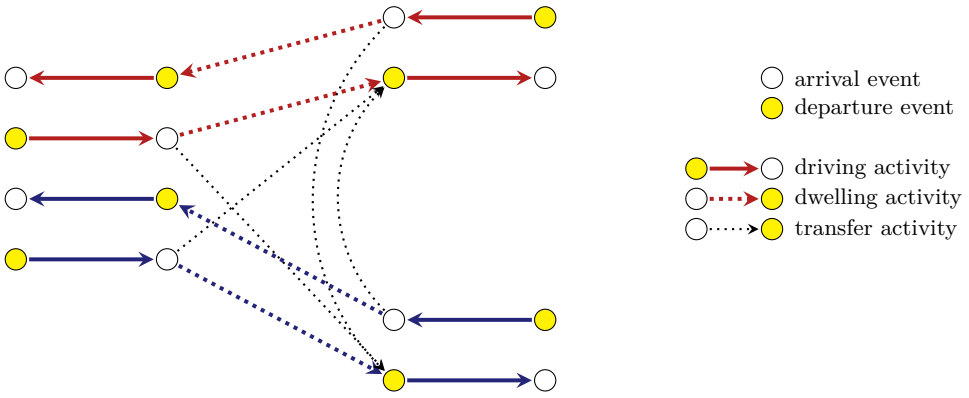

Figure 1 Example of an event-activity network: A bifurcation of a red and a blue line with a few passenger transfers.

A passenger journey in a public transportation network has a natural interpretation as a path in a classical event-activity network. However, when a timetable and hence activity lengths (called *tensions*) have been determined, it is not reasonable to solve a shortest path problem on G as is: Passengers are typically not required to start and end their journey at specific departure or arrival events. Instead, they might choose any line serving a close-by stop within walking distance of their point of origin. Analogously, it makes little sense to select a specific arrival event of a specific line at a specific stop as the endpoint of a journey.

We therefore extend the event-activity network by *source cells* and *target cells* that model the origins and destinations of passengers, respectively. Source cells have no incoming edges and can be connected to several departure events by means of *access activities*. In the same vein, target cells have no outgoing edges, but can be reached from several arrival events by other access activities. Those access activities allow, e.g., to model walking times to different stop locations.

▶ **Definition 1** (Extended Event-Activity Network). *An* extended event-activity network *is a directed graph* $G = (V, A)$ *such that*

$$V = V_{dep} \dot\cup V_{arr} \dot\cup V_{source} \dot\cup V_{target},$$
$$A = A_{access} \dot\cup A_{drive} \dot\cup A_{dwell} \dot\cup A_{transfer} \dot\cup A_{other},$$
$$A_{access} \subseteq (V_{source} \times V_{dep}) \cup (V_{arr} \times V_{target}),$$
$$A_{drive} \subseteq V_{dep} \times V_{arr},$$
$$A_{dwell} \subseteq V_{arr} \times V_{dep},$$

$A_{transfer} \subseteq V_{arr} \times V_{dep}$.

The same construction appears in, for example, [20, 10, 18, 12]. To illustrate, the event-activity network of Figure 1 is extended in Figure 2.

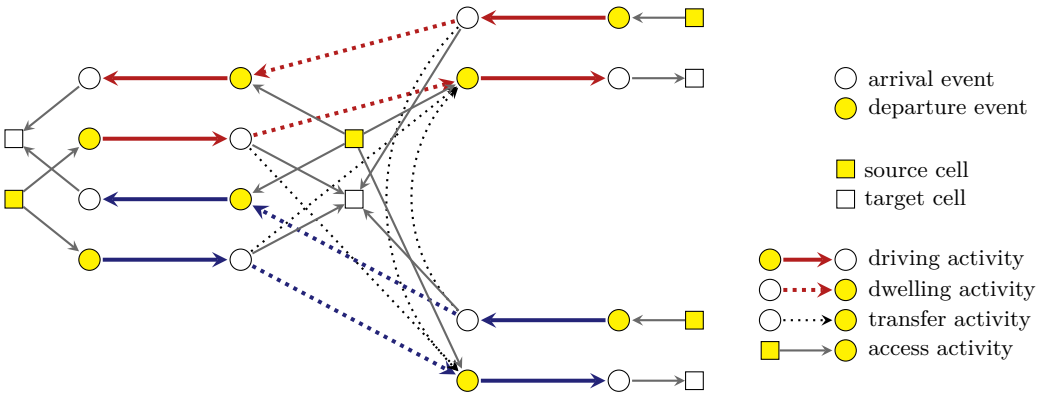

Figure 2 Example of an extended event-activity network, extending Figure 1 by source cells, target cells, and access activities at each station.

We can now interpret passenger routes as special paths in an extended event-activity network G, which connect source cells in V_{source} to target cells in V_{target} such that they begin and end with a respective access activity and their remaining activities come from $A_{\text{drive}} \dot\cup A_{\text{dwell}} \dot\cup A_{\text{transfer}}$. Moreover, if we are given an appropriate time duration for each activity, we can ask for a shortest passenger route from a source cell s to a target cell t, and answer with a shortest s-t-path in G. How to find these times is the subject of the subsequent subsection.

2.2 The TimPass Problem

The TimPass problem is an extension of the *Periodic Event Scheduling Problem* (PESP) for periodic timetabling to extended event-activity networks incorporating passenger routing.

▶ **Definition 2** (TimPass). *Consider a tuple* (G, T, p, ℓ, u, D)*, where*

- $G = (V, A)$ *is an extended event-activity network,*
- $T \in \mathbb{N}$ *is a* period time*,*
- $\rho \in \mathbb{R}_{\geq 0}$ *is a* transfer penalty*,*
- $\ell \in \mathbb{R}^A_{\geq 0}$ *are* lower bounds *on activity lengths,*
- $u \in \mathbb{R}^A_{\geq 0}$ *with* $0 \leq u - \ell < T$ *are* upper bounds *on activity lengths,*
- $D = (d_{st}) \in \mathbb{R}^{V_{source} \times V_{target}}_{\geq 0}$ *is an* origin-destination (OD) matrix.

The Integrated Periodic Timetabling and Passenger Routing Problem *(TimPass) is to find a* periodic timetable $\pi \in [0, T)^{V_{dep} \cup V_{arr}}$ *and a* periodic tension $x \in \mathbb{R}^A$ *such that*

- $\pi_j - \pi_i \equiv x_a \bmod T$ *for all* $a = (i, j) \in A \setminus A_{access}$*,*
- $\ell_a \leq x_a \leq u_a$ *for all* $a \in A \setminus A_{access}$*,*
- *the* total travel time $\sum_{(s,t) \in V_{source} \times V_{target}} d_{st} \tau_{st}$ *is minimum, where* τ_{st} *is the cost of a shortest* s-t*-path in* G *with respect to activity costs* c^x *given by*

$$c^x_a := \begin{cases} \ell_a & \text{if } a \in A_{access}, \\ x_a + \rho & \text{if } a \in A_{transfer}, \\ x_a & \text{otherwise}, \end{cases} \qquad \text{for all } a \in A. \tag{1}$$

The periodic timetable π gives the departure and arrival times, which repeat with a period time of T. The periodic tension x captures the length of each activity $a = (i, j) \in A \setminus A_{\text{access}}$ and is determined up to an integer multiple of T by the event potentials π_i and π_j such that $\ell \leq x \leq u$ holds. Since the duration of access activities models walking times, we consider their tensions fixed to their lower bounds. To route the passengers, we assume that for each OD pair $(s, t) \in V_{\text{source}} \times V_{\text{target}}$, all d_{st} passengers travel along the same shortest path. This path is determined based on the travel times derived from the periodic tension x and the lower bounds of the access activities. Additionally, each transfer activity incurs a penalty, with its actual duration increased by a supplement of ρ.

▶ Remark 3. Since determining the existence of a feasible periodic timetable is already NP-hard [14], this property naturally extends from PESP to TimPass.

It is straightforward to state a mixed-integer programming formulation for TimPass:

$$\text{Minimize} \quad \sum_{a \in A \setminus A_{\text{access}}} w_a x_a + \sum_{a \in A_{\text{transfer}}} \rho w_a + \sum_{a \in A_{\text{access}}} \ell_a w_a \tag{2}$$

$$\text{subject to} \quad \pi_j - \pi_i + T z_a = x_a, \qquad a = (i, j) \in A \setminus A_{\text{access}}, \tag{3}$$

$$\ell_a \leq x_a \leq u_a, \qquad a \in A \setminus A_{\text{access}}, \tag{4}$$

$$0 \leq \pi_i \leq T - 1, \qquad i \in V_{\text{dep}} \cup V_{\text{arr}}, \tag{5}$$

$$z_a \in \mathbb{Z}, \qquad a \in A \setminus A_{\text{access}} \tag{6}$$

$$w_a = \sum_{(s,t) \in \mathcal{D}} \sum_{p \in P_{st} : a \in p} d_{st} f_p, \qquad a \in A, \tag{7}$$

$$\sum_{p \in P_{st}} f_p = 1, \qquad (s, t) \in \mathcal{D}, \tag{8}$$

$$f_p \in \{0, 1\}, \qquad p \in P_{st}, (s, t) \in \mathcal{D} \tag{9}$$

Here, we write $\mathcal{D} \coloneqq \{(s, t) \in V_{\text{source}} \times V_{\text{target}} \mid d_{st} > 0\}$ for the set of OD pairs with positive demand. The constraints (3)–(6) define PESP, while (8)–(9) describe the selection of exactly one s-t-path per OD pair $(s, t) \in \mathcal{D}$ from the set of all such paths P_{st}. The number of passengers w_a on each activity $a \in A$ is then simply the sum over all passengers whose selected s-t-path contains a, as dictated by constraint (7). The objective (2) is to minimize the total (perceived, if $\rho > 0$) travel time of all passengers.

▶ Remark 4. A solution of (2)–(9) can be recovered in polynomial time from a periodic timetable π: For each $a = (i, j) \in A$, set $x_a \coloneqq (\pi_j - \pi_i - \ell_a) \bmod T + \ell_a$. Alternatively, π can be reconstructed from x by a graph traversal [9]. The path variables f_p can be obtained by computing shortest s-t-paths for all $(s, t) \in \mathcal{D}$.

▶ Remark 5. There are several ways to reformulate the program (2)–(9). For instance, the Periodic Event Scheduling Problem admits formulations using integral cycle bases, or time expansion. Likewise, the TimPass model can be adapted. For example, the constraint (9) may also be replaced by $f_p \geq 0$, and there is also a time-expanded version.

3 The Restricted Integrated Modulo Network Simplex Algorithm

In this paper, we will not focus on mixed-integer programming techniques to solve TimPass, but rather on heuristic combinatorial algorithms. In this section, we recall the *integrated modulo network simplex algorithm* as introduced in [10].

The *modulo network simplex* (MNS) method adapts the well-known network simplex for periodic timetabling [13]. Feasible solutions of PESP (constraints (3)–(6)) can be encoded by *spanning tree structures* $\mathcal{T} = \mathcal{T}_\ell \dot\cup \mathcal{T}_u$, which are spanning trees of the event-activity network where tree activity tensions are fixed to either their lower or upper bound. We can explore the solution space from a given initial timetable by iteratively adding a co-tree activity to the tree and removing one of the activities along the induced fundamental cycle. This yields a neighborhood relation between spanning tree structures. The method terminates, generally in a local optimum, if neither any pivot operation nor shifting of event timings along special cuts yields an improved objective value.

Let G be an extended event-activity network. The association between solutions and spanning tree structures on $G[V_{\text{dep}} \cup V_{\text{arr}}]$ holds for TimPass [2] as well, we merely have to decide when the passenger paths are updated throughout the procedure. One approach is to simply compute the shortest s-t-path for every OD pair $(s,t) \in \mathcal{D}$ once stuck in a local optimum, or after every pivot operation. We have found previously however that this approach does not perform significantly better than solving PESP with fixed passenger paths and then computing the shortest paths of the final timetable [10].

Instead, when we determine the next improving pivot operation and compare the objective value of the current tree structure to each of its neighbors, we have to examine the neighbors under their respective optimal passenger paths. Since the number of co-tree activities whose tensions are altered by the pivot operation can be arbitrarily large, there is no obvious way to tell which shortest passenger paths are different in the neighboring solution. Therefore, for every pivot candidate that results in a feasible timetable, we have to run Dijkstra's algorithm for every cell $s \in V_{\text{source}}$. We call this method the *integrated modulo network simplex* (IMNS). It requires $\mathcal{O}(|\mathcal{T}| \cdot |A \setminus (A_{\text{access}} \cup \mathcal{T})| \cdot |V_{\text{source}}|) = \mathcal{O}(|V_{\text{dep}} \cup V_{\text{arr}}| \cdot |A \setminus A_{\text{access}}| \cdot |V_{\text{source}}|)$ executions of Dijkstra's algorithm to obtain shortest passenger path trees in each iteration. Clearly, this is computationally intractable on large instance sizes.

To address this, we have previously devised the *restricted integrated modulo network simplex* (RIMNS) [10]. For a given k, we first compute the k shortest paths using at most two transfers for every OD pair $(s,t) \in \mathcal{D}$ assuming every activity has its tension at the lower bound. If every connection between s and t requires at least three transfers, we store only a single shortest path. When examining the pivot candidates, we simply select whichever path is shortest under the resulting activity tensions from each OD pair's current path pool. After the pivot operation, we compute the actual shortest paths, update the objective if needed, and add any new s-t-path to the respective pool. Note that it suffices to store access and transfer activities to fully encode a passenger path due to the structure of extended event-activity networks. Moreover, we have empirically determined $k = 20$ to offer the best trade-off between runtime penalty and solution quality impact [10]. We use this method as a benchmark in our computational study presented in Section 6.

4 Geometric Interpretation of TimPass

As seen in the last section, the core question in the integrated modulo network simplex algorithm is when to compute a new routing of the passenger paths. We will now gain insight on this matter from a geometric point of view, developing a new heuristic for TimPass. To do so, we begin with a minimalistic example.

4.1 Introductory Examples

▶ **Example 6** (Slow line and express line). We consider a TimPass instance as shown in Figure 3. We take $T = 20$ and assume that all but the two transfer activities (dotted arrows) are fixed, i.e. $\ell_a = u_a$, and that there are no transfer penalties. In this network, there are only two possible passenger paths from s to t: The path p_r uses the red slow line, the other path p_b includes the blue express line. Let x_1 and x_2 denote the periodic tensions of the two transfer arcs, respectively. We make the following observation: The path p_r is a shortest s-t-path if and only if f $x_1 + x_2 \geq 15$, and p_b is a shortest s-t-path if and only if $x_1 + x_2 \leq 15$.

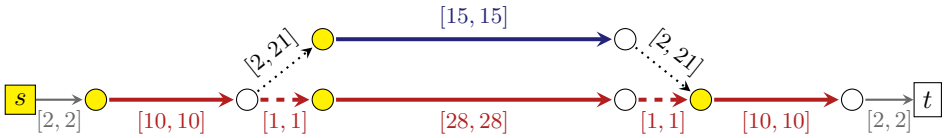

Figure 3 Extended event-activity network for Example 6, with labels $[\ell_a, u_a]$ for each activity a.

For the timetabling part, we note that in this instance, a periodic tension is fully determined by x_1 and x_2. However, feasible periodic tensions are only those where both paths take the same time modulo T due to the cycle periodicity property [9, 14]. When depicting the possible values of x_1 and x_2 according to their bounds as the square $[2, 21] \times [2, 21]$, we hence identify the two blue highlighted line segments in Figure 4 as the space of feasible periodic tensions.

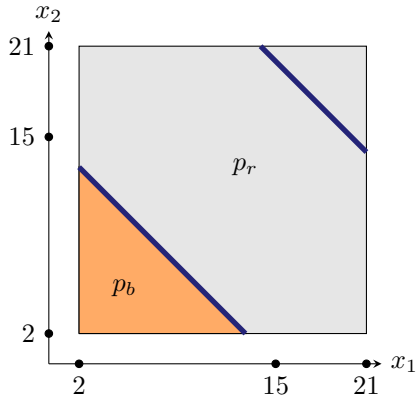

Figure 4 Geometry of Example 6. For clarity, note that the extension of the lower left blue line segment intersects the axes in $(0, 15)$ and $(15, 0)$, while the extension of the upper right blue line segment intersects in $(0, 35)$ and $(35, 0)$. This corresponds to setting the tension of either transfer activity to 0 which violates the bounds $[2, 21]$ and is hence outside the feasible region.

The hyperplane $x_1 + x_2 = 15$ divides the square into two polytopes: For (x_1, x_2) on the bottom-left side (orange in Figure 4), p_b is shorter than p_r, and on the top-right side (grey in Figure 4), p_r is shorter than p_b.

We conclude from Figure 4 that any (x_1, x_2) with $x_1 + x_2 = 15$ is an optimal solution to the TimPass instance: In this case, p_r and p_b provide the same travel time and taking the blue express line can never be faster anyway, since we need to transfer back to the red slow line to reach cell t.

In a geometric language, we can make the following observations in Figure 4:

1. The space of feasible periodic tensions is a disjoint union of line segments in the square.
2. The hyperplane $x_1 + x_2 = 15$ subdivides the square into two polytopes, where each polytope corresponds to the region where one of the two paths is a shortest path.
3. A line segment of feasible tensions is never part of the interior of more than one shortest path region.

▶ **Example 7** (Two lines). We now turn to a TimPass instance with an extended event-activity network as in Figure 5. Again, there is a red line and a blue line, but we can reach t from s using three paths: p_{bb} (blue-blue), p_{br} (blue-red), p_{rr} (red-red). The only interesting tensions are x_{br} for the transfer from blue to red and x_{rr} for the dwelling activity of the red line. The shortest s-t path is

$$\begin{aligned} &p_{bb} \text{ if } x_{br} \geq 10 \text{ and } x_{rr} \geq 10, \\ &p_{br} \text{ if } x_{br} \leq 10 \text{ and } x_{br} \leq x_{rr}, \\ &p_{rr} \text{ if } x_{rr} \leq 10 \text{ and } x_{rr} \leq x_{br}. \end{aligned} \tag{10}$$

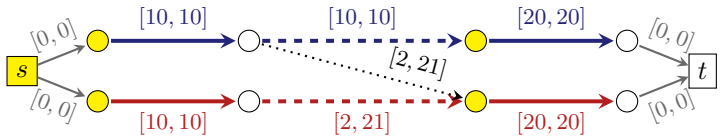

Figure 5 Extended event-activity network for Example 7, with labels $[\ell_a, u_a]$ for each activity a.

In Figure 6, we illustrate this situation: By the lower and upper bounds, (x_{br}, x_{rr}) lives again in the square $[2, 21] \times [2, 21]$. The square is now divided into three regions according to (10). In this example, any combination of (x_{br}, x_{rr}) determines a feasible periodic tension: The event-activity network without the access activities forms a tree.

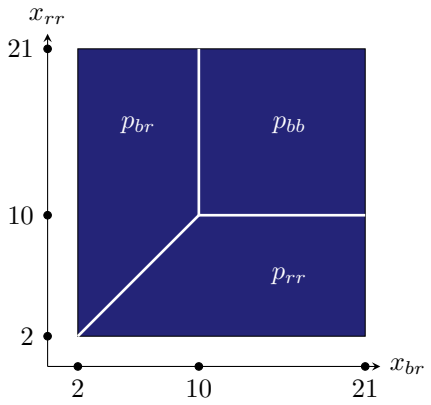

Figure 6 Geometry of Example 7.

Revisiting the three geometric observations from Example 6, we note for Example 7:

1. The space of feasible tension is the full square.
2. The three paths give rise to a polyhedral subdivision of the square, each maximal region corresponding to the unique shortest s-t-path with respect to the periodic tensions of that region.

3. The shortest path subdivision is also a proper subdivision of the square of periodic tensions.

Our interpretations look different at first glance. We offer a unifying perspective in the next subsection.

4.2 Theoretic Results

We will now generalize the geometric observations made in the previous examples. Let $I = (G, T, \rho, \ell, u, D)$ be a TimPass instance.

▶ **Definition 8** (Periodic Tension Spaces). *The* fractional periodic tension polytope *of I is*

$$X_{LP} \coloneqq \prod_{a \in A \setminus A_{access}} [\ell_a, u_a].$$

The periodic tension space *of I is*

$$X \coloneqq \{x \in X_{LP} \mid \exists \pi \in [0,T)^{V_{dep} \cup V_{arr}} \ \forall a = (i,j) \in A \setminus A_{access} : \pi_j - \pi_i \equiv x_a \bmod T\}.$$

The fractional tension polytope is the space of all vectors $x \in \mathbb{R}^{A \setminus A_{\text{access}}}$ that satisfy the constraints of the linear programming relaxation of (2)–(9). X_{LP} is by definition a hyperrectangle, such as the squares in Example 6 and Example 7. The periodic tension space X is the space of vectors x that are feasible for the mixed-integer program (2)–(9), e.g., the blue line segments in Figure 4 and the full square in Figure 6.

▶ **Definition 9** (Polytrope [5]). *For each $z \in \mathbb{Z}^A$, we define the* polytrope

$$R(z) \coloneqq \{x \in X_{LP} \mid \exists \pi \in \mathbb{R}^{V_{dep} \cup V_{arr}} \ \forall a = (i,j) \in A \setminus A_{access} : \pi_j - \pi_i + T z_a = x_a\}.$$

We refer to [7] for the origin of the name *polytrope*, and recall a few results from [5]: Let Γ be the cycle matrix of an integral cycle basis of the induced subgraph $G[V_{\text{dep}} \cup V_{\text{arr}}]$.

1. For $z, z' \in \mathbb{Z}^A$ we have either $R(z) \cap R(z') = \emptyset$ or $R(z) = R(z')$, the latter case occurring if and only if $\Gamma z = \Gamma z'$.
2. The periodic tension space is the union of all $R(z)$, taken over all $z \in \mathbb{Z}^A$.
3. For each $z \in \mathbb{Z}^A$, solving PESP restricted to $x \in R(z)$ is a linear program dual to minimum cost network flow, and hence solvable in polynomial time.
4. One can define a neighborhood relation such that two non-empty polytropes $R(z)$ and $R(z')$ are neighbors if Γz and $\Gamma z'$ differ by $\pm$ a column of Γ.

These insights led to *tropical neighborhood search*, a powerful heuristic for the Periodic Event Scheduling Problem [4]. This is a local search that starts with a non-empty polytrope and scans for improving neighbors. Any polytrope has at most $2|A \setminus A_{\text{access}}|$ neighbors, and PESP can be solved on each polytrope in polynomial time.

We will generalize tropical neighborhood search to *integrated tropical neighborhood search* (ITNS). To this end, we need to include passenger paths into our considerations.

▶ **Definition 10** (Shortest Path Subdivision). *Let $s \in V_{source}$, $t \in V_{target}$ and let p be an s-t-path in G. We define*

$$S_{s,t}(p) \coloneqq \{x \in X_{LP} \mid c(x)(p) \leq c(x)(p') \text{ for all } s\text{-}t\text{-paths } p'\},$$

where c^x is the cost function defined in (1)*. The collection of $S_{s,t}(p)$ gives rise to the* shortest path subdivision $\mathcal{S}_{s,t}$ *of X_{LP}.*

Speaking geometrically, finding a passenger routing for a periodic tension $x \in X$ then amounts to determining, for each $(s,t) \in \mathcal{D}$, an s-t-path p such that $x \in S_{s,t}(p)$. In Figure 4, the orange polytope is $S_{s,t}(p_b)$, defined by the inequality $x_1 + x_2 \leq 15$, and the grey polytope is $S_{s,t}(p_r)$, defined by $x_1 + x_2 \geq 15$. Analogously, we see the subdivision of the square X_{LP} into $S_{s,t}(p_{bb}), S_{s,t}(p_{br}), S_{s,t}(p_{rr})$ in Figure 6 according to (10).

The shortest path subdivision $\mathcal{S}_{s,t}$ of X_{LP} naturally induces a subdivision of each polytrope $R(z)$. Looking at Figure 4, this subdivision is rather trivial, as is confirmed by the following:

▶ **Lemma 11.** *Let $u \in V_{dep}$, $v \in V_{arr}$, $z \in \mathbb{Z}^A$. Then p is a shortest u-v-path w.r.t. c^x for some $x \in R(z)$ if and only if it is a shortest u-v-path w.r.t. c^z, where*

$$c_a^z \coloneqq \begin{cases} z_a + \frac{\rho}{T} & \text{if } a \in A_{transfer}, \\ z_a & \text{otherwise} \end{cases} \qquad \text{for all } a \in A \setminus A_{access}.$$

Proof. Let p be a u-v-path. By the construction of extended event-activity networks, p cannot contain access activities. For $x \in R(z)$, we find a periodic timetable π such that $x_a = \pi_j - \pi_i + T z_a$ for all $a = (i,j) \in A \setminus A_{\mathrm{access}}$, and therefore

$$\begin{aligned} c^x(p) &= \sum_{a \in p \cap (A \setminus A_{\mathrm{access}})} x_a + \sum_{a \in p \cap A_{\mathrm{transfer}}} \rho \\ &= \sum_{a=(i,j) \in p \cap (A \setminus A_{\mathrm{access}})} (\pi_j - \pi_i + T z_a) + \sum_{a \in p \cap A_{\mathrm{transfer}}} \rho \\ &= \pi_v - \pi_u + T \sum_{a=(i,j) \in p \cap (A \setminus A_{\mathrm{access}})} z_a + \sum_{a \in p \cap A_{\mathrm{transfer}}} \rho \\ &= \pi_v - \pi_u + T c^z(p). \end{aligned}$$

Therefore, if p and p' are u-v-paths, then $c^x(p) \leq c^x(p')$ holds if and only if $c^z(p) \leq c^z(p')$. ◀

The consequences of Lemma 11 are remarkable: For all tensions inside a polytrope $R(z)$, the same path can be chosen for a shortest path, as the costs depend only on z – as long as the path starts at a departure event and ends at an arrival event. This result naturally extends to paths from source to target cells, where each of them is connected to a single event only, as is the case in Example 6. However, Example 7 demonstrates that this is no longer true when cell nodes are adjacent to multiple events. For a vertex $u \in G$, we define its out-neighborhood $N^+(u) = \{v \in V \mid (u,v) \in A\}$ and its in-neighborhood $N^-(u) = \{v \in V \mid (v,u) \in A\}$.

▶ **Theorem 12.** *Let $s \in V_{source}$ and $t \in V_{target}$ and $z \in \mathbb{Z}^A$. Then $\mathcal{S}_{s,t}$ induces a polyhedral subdivision of $R(z)$. Each maximal region of the subdivision is parametrized by a departure event $u \in N^+(s)$, an arrival event $v \in N^-(t)$, and given by $S_{s,t}(p) \cap R(z)$ for a shortest s-t-path p containing the access activities (s,u) and (v,t).*

Proof. Let $x \in R(z)$, $u \in N^+(s)$, $v \in N^-(t)$, and let p be an s-t-path using (s,u) and (v,t). Then

$$c^x(p) = \ell_{su} + \ell_{vt} + \sum_{a \in p \cap (A \setminus A_{\mathrm{access}})} c_a^x. \tag{11}$$

As in the proof of Lemma 11,

$$c^x(p) = \ell_{su} + \ell_{vt} + \pi_v - \pi_u + T \sum_{a \in p \cap (A \setminus A_{\mathrm{access}})} c_a^z. \tag{12}$$

In particular, the cost c^x of a path p depends only on z, its first departure event, and its last arrival event. If p' is another s-t-path that contains (s,u) and (v,t) as well, then $c^x(p) \leq c^x(p')$ if and only if $c^z(p) \leq c^z(p')$. The maximal regions of $R(z)$ induced by the shortest path subdivision $\mathcal{S}_{s,t}$ are hence described by a pair $(u,v) \in N^+(s) \times N^-(t)$ such that a shortest s-t-path w.r.t. c^x contains both (s,u) and (v,t) for all x in that region. Due to Lemma 11, this shortest path can be chosen to be the same in each such region, say $p_{u,v}$. The region $S_{s,t}(p_{u,v}) \cap R(z)$ is then described by the inequalities

$$c^x(p_{u,v}) \leq c^x(p_{u',v'}) \quad \text{for all } (u',v') \in N^+(s) \times N^-(t) \setminus \{(u,v)\}. \qquad \blacktriangleleft$$

▶ **Corollary 13.** *For a single OD pair $(s,t) \in V_{source} \times V_{target}$ and an arbitrary $z \in \mathbb{Z}^A$, a shortest s-t-path w.r.t. c^x among all $x \in R(z)$ can be found in polynomial time.*

Proof. A tension $x \in R(z)$ can be found in polynomial time by the discussion following Definition 9. By Theorem 12, the shortest path subdivision $\mathcal{S}_{s,t}$ subdivides $R(z)$ into at most $|N^+(s)| \cdot |N^-(t)|$ maximal polyhedral regions. Using Lemma 11, for each such region $R_{u,v}$ labeled by u and v, we can compute a representing path $p_{u,v}$ such that $R_{u,v} = S_{s,t}(p_{u,v}) \cap R(z)$ by computing a shortest path u-v-path w.r.t. c^z, and adding the activities (s,u) and (v,t). We then determine an optimal tension $x^{u,v} \in S_{s,t}(p_{u,v}) \cap R(z)$ by finding an optimal solution x to the linear program

$$\text{Minimize} \quad \sum_{a \in p_{uv} \cap (A \setminus A_{\text{access}})} d_{st} x_a \tag{13}$$

$$\text{subject to} \quad \pi_j - \pi_i + T z_a = x_a, \qquad a = (i,j) \in A \setminus A_{\text{access}}, \tag{14}$$

$$\ell_a \leq x_a \leq u_a, \qquad a \in A \setminus A_{\text{access}}, \tag{15}$$

$$\pi_i \in \mathbb{R}, \qquad i \in V_{\text{dep}} \cup V_{\text{arr}}. \tag{16}$$

This linear program is the restriction of (2)–(9) to the fixed periodic offset z and the fixed path p_{uv}, where we dropped the now constant summand

$$\sum_{a \in p_{u,v} \cap A_{\text{transfer}}} d_{st}\rho + d_{st}\ell_{su} + d_{st}\ell_{vt}$$

in the objective function, and enlarged the domain of π according to the definition of $R(z)$. It remains to select the best tension among the $x^{u,v}$ for all $(u,v) \in N^+(s) \times N^-(t)$. $\blacktriangleleft$

The method in the proof of Corollary 13 is based on the polynomial number of maximal regions of the shortest path subdivision of $R(z)$. For more than one OD pair, this number however becomes exponential, as the regions of the common refinement of all shortest path subdivisions $\mathcal{S}_{s,t}$ are

$$R(z) \cap \bigcap_{(s,t) \in V_{\text{source}} \times V_{\text{target}}} S_{s,t}(p_{u_{s,t},v_{s,t}}) \tag{17}$$

for all combinations of $(u_{s,t}, v_{s,t}) \in N^+(s) \times N^-(t)$ for all $(s,t) \in V_{\text{source}} \times V_{\text{target}}$. These regions single out those periodic tensions, where the same set of passenger routes yields shortest paths for all OD pairs.

When considering all OD pairs, we can hence not expect a polynomial-time algorithm for TimPass on $R(z)$. However, on a single region of the common refinement of all shortest path subdivisions, there is a positive result.

▶ **Corollary 14.** *Let $z \in \mathbb{Z}^A$ and let R be a maximal region of the common refinement of all shortest path subdivisions $\mathcal{S}_{s,t}$ for all OD pairs (s,t). Suppose that R is described in terms of $(u_{s,t}, v_{s,t}) \in N^+(s) \times N^-(t)$ for all $(s,t) \in V_{source} \times V_{target}$. Then TimPass can be solved in polynomial time when x is restricted to $R(z) \cap R$.*

Proof. We determine $p_{u_{s,t},v_{s,t}}$ for all (s,t) as in the proof of Corollary 13. We then solve the restriction of (2)–(9) to the fixed z and the chosen paths, which boils down to solving the linear program (14)–(16) w.r.t. to the objective function

$$\sum_{a \in A \setminus A_{\text{access}}} w_a x_a, \quad \text{where } w_a := \sum_{(s,t) \in \mathcal{D}:\, a \in p_{u_{s,t},v_{s,t}}} d_{st}. \qquad \blacktriangleleft$$

5 Integrated Tropical Neighborhood Search

We now turn back to the question of when to reroute passengers in an alternating timetabling-routing procedure such as the integrated modulo network simplex (cf. Section 3). The geometric answer from Section 4 is the following: The passenger routing can be chosen to be the same on each of the regions (17). However, this investigation is limited to a single polytrope $R(z)$. In our upcoming local search algorithm, *integrated tropical neighborhood search* (ITNS), we will therefore always re-route the passengers when we leave a polytrope. Inside a polytrope, we will either find a solution heuristically using Corollary 14, or be more extensive and scan for local improvements by modifying one OD pair at a time only (Corollary 13).

We outline the three components of ITNS on a high level.

5.1 The Coarse Polytrope Heuristic

In the coarse polytrope heuristic, for a given initial tension $x \in R(z)$, we determine the region R in the sense of (17) such that $x \in R \cap R(z)$ and solve TimPass optimally by means of Corollary 14. To this end, we solve the linear program indicated in the proof, and its optimal objective value will give the minimum total travel time on $R \cap R(z)$.

▶ **Algorithm 15** (Coarse Polytrope Heuristic)**.**
Input: periodic tension $x \in R(z)$
Output: periodic tension $x^* \in R(z)$ and its total travel time τ^*

1. For all $(s,t) \in \mathcal{D}$, compute shortest paths $p_{s,t}$ w.r.t. c^x, giving $(u_{s,t}, v_{s,t}) \in N^+(s) \times N^-(t)$.
2. Solve TimPass on (17) via Corollary 14 to obtain a tension $x^* \in R(z)$ and its total travel time τ^*. Return x^* and τ^*.

5.2 The Fine Polytrope Heuristic

The fine polytrope heuristic proceeds as the coarse heuristic, but then tries to improve the solution by re-routing passengers for single OD pairs in the spirit of Corollary 13.

▶ **Algorithm 16** (Fine Polytrope Heuristic)**.**
Input: periodic tension $x \in R(z)$
Output: periodic tension $x^* \in R(z)$ and its total travel time τ^*

1. Obtain $x^* \in R(z) \cap \bigcap_{(s,t) \in V_{\text{source}} \times V_{\text{target}}} S_{s,t}(p_{s,t})$ from Algorithm 15.
2. For a list of OD pairs (s,t):

- Enumerate the regions $S_{s,t}(p_{u,v})$ be computing representing s-t-paths $p_{u,v}$ as in the proof of Corollary 13.
- For each such region, replace $p_{s,t}$ by $p_{u,v}$ and solve TimPass on the new region via Corollary 14.
- If a solution with better travel time has been found, set $p_{s,t} \coloneqq p_{u,v}$ and update x^*.

3. Return x^* and its total travel time τ^*.

In view of Theorem 12, Step (2) can be understood as a heuristic exploration of the neighboring regions of $R(z) \cap \bigcap_{(s,t)\in V_{\text{source}}\times V_{\text{target}}} S_{s,t}(p_{s,t})$. The algorithm may be fine-tuned by adapting the selection and sorting of the OD pairs. Note that for the linear programs in (2), only the objective function changes, so that warm-starting is possible.

5.3 Integrated Tropical Neighborhood Search

We embed the two polytrope heuristics into tropical neighborhood search for periodic timetabling [4].

▶ **Algorithm 17** (Integrated Tropical Neighborhood Search, ITNS).
Input: feasible TimPass instance
Output: periodic tension $x^* \in R(z)$ and its total travel time τ^*

1. Compute a feasible solution to TimPass, giving $x \in R(z)$.
2. Obtain $x^z \in R(z)$ and τ^z by Algorithm 15 or Algorithm 16.
3. Use Algorithm 15 to compute $\tau^{z'}$ with tension $x^{z'} \in R(z')$ among all neighbors $R(z')$ of $R(z)$.
4. If there is no z' with $\tau^{z'} < \tau^z$, return x^z and τ^z.
5. Choose a z' with $\tau^{z'} < \tau^z$, set $z \coloneqq z'$ and go to Step (2).

As computing shortest paths for all OD pairs is the major bottleneck, we do not use the fine polytrope heuristic in Step (3). Depending on the algorithm chosen in Step (2), we speak of *coarse* or *fine* ITNS. The ITNS might be refined in several ways, e.g., by different pivot or quality-first rules, see [4].

6 Computational Study

We present novel computational results of applying the (restricted) integrated modulo network simplex and the coarse/fine integrated tropical neighborhood search to the TimPassLib benchmark library [17].

6.1 Obtaining an Initial Solution

All presented algorithms in this paper are neighborhood searches that require an initial feasible timetable. We outline a heuristic to obtain those by constructing a spanning tree structure $\mathcal{T} = \mathcal{T}_\ell \dot\cup \mathcal{T}_u$. First, note that it is usually assumed that $u_a = \ell_a + T - 1$ holds for transfer activities $a \in A_{\text{transfer}}$. So if $A_{\text{other}} = \varnothing$, adding all driving and dwelling activities to the lower bound tree $\mathcal{T}_\ell$ and connecting the resulting line components by transfer activities always yields a feasible timetable. We can weigh the transfer activities by the number of passengers on them subject to a lower bound routing, i.e., with respect to shortest paths when all tensions x_a are at their lower bound ℓ_a. If there are any other activities in A_{other}, then special care has to be taken to ensure feasibility:

- *Turnaround* activities model the required downtime between the end of a line and the beginning of its reversal direction due to constraints such as driver breaks. Thus, the driving and dwelling activities of a line, its reversal direction and turnover activities form a cycle whose tensions have to sum to zero modulo the period time T due to the cycle periodicity property. We always add this cycle to the tree omitting one of its activities. Some activities may have to have their tension fixed to the upper bound to ensure feasibility.
- If a line is repeated within the global period of the instance, its events are copied appropriately often in the event-activity network and connected by *synchronization* activities. We construct the connected tree component of a line to always include all of its repetitions along those timing activities.
- Like transfers, *headway* activities connect line components but may have bounds that impact feasibility. When we add a line component to the tree under consideration of the previous two points, we collect all outgoing transfer and headway activities. We then process them in some order to either connect additional lines to the growing tree or, if both events incident to the activity are already in the tree, we store them for later. Once we have connected all lines and the tree is spanning, we check if the tension of the remaining co-tree activities is feasible. If not, we attempt pivot operations and to switch up membership in $\mathcal{T}_\ell$ and $\mathcal{T}_u$ along the fundamental cycle until we have a feasible timetable, or we have to give up.

Whether this process succeeds in finding a feasible timetable is sensitive to the order in which we process transfer and other activities. It always succeeds on the TimPassLib instances if we process activities in A_{other} in decreasing order of $u_a - \ell_a$ and then the transfer activities weighted by the lower bound routing, or, if this fails, in increasing order of $u_a - \ell_a$ until $u_a - \ell_a > T/2$, then the weighted transfer activities, then the remaining other activities.

6.2 Computational Results

We ran C++ implementations of our proposed methods on an initial solution obtained as explained in the previous subsection on each TimPassLib instance with a time limit of an hour. For an overview on the instance sizes we refer to [17] and `https://timpasslib.aalto.fi/`. For ITNS, we report the best solutions in terms of total travel time found across three parameter settings concerning a quality-first rule and numbers of OD pairs considered in Step (3) of Algorithm 16. We always sort the OD pairs with respect to decreasing weighted slack along the current path p, i.e., $d_{st} \sum_{a \in p}(x_a - \ell_a)$, so that heavy demand relations with much longer travel times than necessary come first.

All computations have been carried out on an Intel Core i7-9700K CPU with 64 GB RAM. The results are presented in Table 1 in the appendix.

For six of the instances, we provide better incumbent solutions (Hamburg, grid, long, metro, Erding20, Erding21). The Stuttgart instance is very large and causes a memory problem in the ITNS implementation. The RIMNS performs particularly strong. For ITNS, the coarse version is on level with RIMNS, better on larger instances, but worse on smaller.

The fine ITNS is never better than coarse ITNS. This is also highlighted in Table 2 in the appendix, where we collect the computation times and the total travel time improvement in relation to the initial solution. It turns out – see that last column of Table 2 – that the improvements made in (2) of Algorithm 16 are very minor, while moving to neighboring polytropes has a larger impact on the total travel time. In particular, it is not advisable to spend computation time for the fine ITNS.

Unfortunately, we could not determine better travel times for the RxLy instances within one hour, whose current incumbents are based on a heavy machinery of periodic timetable optimization [3, 17]. However, since our methods performed strongly on the instances where the time limit was not an issue, we became curious if we would be able to beat the best known incumbents if we allotted more computation time. Moreover, it is straightforward to parallelize the exploration of the neighborhood of the current solution for both MNS and TNS, and the shortest path computations for TNS, yielding massive speed-up by moving to high-throughput cluster nodes with 96 threads composed of Intel Xeon Gold 6342 CPUs with 512 GB RAM. It becomes apparent that the RIMNS outperforms coarse ITNS in this setting, the latter stopping earlier with worse local optima. Nevertheless, within a wall time limit of 24 hours, we provide better incumbent solutionsfor all TimPassLib instances except toy, toy2, and Stuttgart, see Table 3 in the appendix. Note that toy and toy2 have already been solved to proven optimality previously.

7 Conclusion and Outlook

We have analyzed the geometry of integrated periodic timetabling and passenger routing, culminating in an extension of tropical neighborhood search to an integrated setting. The ITNS heuristic is complementary to the IMNS algorithm family, and is of a similar computational power. Both are promising candidates to get a fast good-quality solution for TimPass problems, as is demonstrated by our new incumbent solutions for the TimPassLib instances.

What remains open on the theory side is to settle the complexity status of optimizing TimPass over a single polytrope for all OD pairs simultaneously. Another line of future work, more on the practical side, would be to integrate the path restriction techniques from RIMNS into ITNS. In general, we believe that including ITNS into larger frameworks for solving TimPass instances will prove useful, and that further algorithmic refinements and implementation improvements are possible.

References

1 Ralf Borndörfer, Heide Hoppmann, and Marika Karbstein. Passenger routing for periodic timetable optimization. *Public Transport*, 9(1):115–135, 2017. doi:10.1007/s12469-016-0132-0.

2 Ralf Borndörfer, Heide Hoppmann, Marika Karbstein, and Fabian Löbel. The Modulo Network Simplex with Integrated Passenger Routing. In Andreas Fink, Armin Fügenschuh, and Martin Josef Geiger, editors, *Operations Research Proceedings 2016*, pages 637–644. Springer International Publishing, 2017. doi:10.1007/978-3-319-55702-1_84.

3 Ralf Borndörfer, Niels Lindner, and Sarah Roth. A concurrent approach to the periodic event scheduling problem. *Journal of Rail Transport Planning & Management*, 15:100175, 2020. doi:10.1016/j.jrtpm.2019.100175.

4 Enrico Bortoletto, Niels Lindner, and Berenike Masing. Tropical Neighbourhood Search: A New Heuristic for Periodic Timetabling. In Mattia D'Emidio and Niels Lindner, editors, *22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2022)*, volume 106 of *Open Access Series in Informatics (OASIcs)*, pages 3:1–3:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 2190-6807. doi:10.4230/OASIcs.ATMOS.2022.3.

5 Enrico Bortoletto, Niels Lindner, and Berenike Masing. The Tropical and Zonotopal Geometry of Periodic Timetables. *Discrete & Computational Geometry*, 2024. doi:10.1007/s00454-024-00686-2.

6 Philine Gattermann, Peter Großmann, Karl Nachtigall, and Anita Schöbel. Integrating Passengers' Routes in Periodic Timetabling: A SAT approach. In Marc Goerigk and Renato F. Werneck, editors, *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*, volume 54 of *Open Access Series in Informatics (OASIcs)*, pages 3:1–3:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2016.3.

7 Michael Joswig and Katja Kulas. Tropical and ordinary convexity combined. *Advances in Geometry*, 10(2):333–352, 2010. Publisher: De Gruyter Section: Advances in Geometry. doi:10.1515/advgeom.2010.012.

8 Michael Joswig and Benjamin Schröter. Parametric Shortest-Path Algorithms via Tropical Geometry. *Mathematics of Operations Research*, 47(3):2065–2081, 2022. Publisher: INFORMS. doi:10.1287/moor.2021.1199.

9 Christian Liebchen. *Periodic timetable optimization in public transport.* PhD thesis, Technicsche Universität Berlin, 2006.

10 Fabian Löbel, Niels Lindner, and Ralf Borndörfer. The Restricted Modulo Network Simplex Method for Integrated Periodic Timetabling and Passenger Routing. In Janis S. Neufeld, Udo Buscher, Rainer Lasch, Dominik Möst, and Jörn Schönberger, editors, *Operations Research Proceedings 2019*, pages 757–763, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-48439-2_92.

11 Bernardo Martin-Iradi and Stefan Ropke. A column-generation-based matheuristic for periodic and symmetric train timetabling with integrated passenger routing. *European Journal of Operational Research*, 297(2):511–531, 2022. doi:10.1016/j.ejor.2021.04.041.

12 Berenike Masing, Niels Lindner, and Enrico Bortoletto. Computing All Shortest Passenger Routes with a Tropical Dijkstra Algorithm, 2024. arXiv:2412.14654 [math]. doi:10.48550/arXiv.2412.14654.

13 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 *Open Access Series in Informatics (OASIcs)*, pages 1–15, Dagstuhl, Germany, 2008. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2008.1588.

14 Michiel A. Odijk. Construction of periodic timetables, Part 1: A cutting plane algorithm. Technical Report 94-61, TU Delft, 1994.

15 Gert-Jaap Polinder, Marie Schmidt, and Dennis Huisman. Timetabling for strategic passenger railway planning. *Transportation Research Part B: Methodological*, 146:111–135, 2021. doi:10.1016/j.trb.2021.02.006.

16 Stephanie Riedmüller. A path-based model for integrated periodic timetabling and passenger routing. Master's thesis, Freie Universität Berlin, 2023.

17 Philine Schiewe, Marc Goerigk, and Niels Lindner. Introducing TimPassLib – A Library for Integrated Periodic Timetabling and Passenger Routing. *Operations Research Forum*, 4, 2023. doi:10.1007/s43069-023-00244-1.

18 Philine Schiewe and Anita Schöbel. Periodic Timetabling with Integrated Routing: Toward Applicable Approaches. *Transportation Science*, 54(6):1714–1731, 2020. Publisher: INFORMS. doi:10.1287/trsc.2019.0965.

19 Marie Schmidt. *Integrating Routing Decisions in Public Transportation Problems.* Springer Optimization and Its Applications. Springer New York, NY, 2014. doi:10.1007/978-1-4614-9566-6.

20 Marie Schmidt and Anita Schöbel. Timetabling with passenger routing. *OR Spectrum*, 37(1):75–97, 2015. doi:10.1007/s00291-014-0360-0.

21 Paolo Serafini and Walter Ukovich. A Mathematical Model for Periodic Scheduling Problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989. doi:10.1137/0402049.

22 Michael Siebert and Marc Goerigk. An experimental comparison of periodic timetabling models. *Computers & Operations Research*, 40(10):2251–2259, 2013. doi:10.1016/j.cor.2013.04.002.

A Result Tables

instance	best known bound	best known solution	IMNS	RIMNS	coarse ITNS	fine ITNS
Hamburg	139 892 927	141 629 305	141 610 697	141 610 697	141 842 262	141 842 262
Schweiz	60 084 289	62 622 935	65 322 356	63 899 227	64 910 675	65 014 781
toy	21 466	21 466	21 466	21 466	21 466	21 466
toy2	19 114	19 114	19 123	19 130	19 198	19 198
regional	1 804 642	1 834 884	1 855 092	1 855 092	1 855 092	1 855 092
grid	47 824	49 279	48 894	48 922	49 775	49 787
long	64 906 980	67 480 984	68 300 017	67 468 861	68 296 696	68 296 696
metro	11 978 129	12 019 079	11 997 040	12 029 681	12 029 681	12 029 681
Erding20	12 206 083	12 258 126	12 239 162	12 239 477	12 270 189	12 270 701
Erding21	12 206 083	12 307 765	12 258 531	12 258 986	12 298 178	12 300 257
Stuttgart	45 072 189 100	48 325 440 573	48 724 661 870	48 724 661 870	48 724 661 870	48 724 661 870
R1L1	522 575 407	542 908 145	548 590 857	547 888 106	548 783 948	549 254 243
R1L2	522 212 362	542 381 697	546 637 453	546 637 453	547 430 602	547 793 546
R1L3	522 199 838	543 067 240	547 066 363	547 052 294	547 404 978	548 056 874
R1L4	520 799 059	537 879 494	541 161 369	540 579 945	541 612 283	541 795 513
R2L1	650 575 045	681 061 389	687 561 970	686 752 397	686 026 782	686 571 886
R2L2	650 293 220	676 836 085	687 143 153	687 298 476	685 036 749	686 155 351
R2L3	649 767 761	675 793 893	681 918 089	681 601 445	680 839 415	681 528 634
R2L4	647 184 195	667 537 183	671 675 462	670 364 469	670 698 507	670 824 345
R3L1	665 804 283	694 086 648	704 342 418	702 079 664	702 474 126	703 168 722
R3L2	665 719 574	694 334 373	704 352 020	701 666 941	702 618 923	703 427 280
R3L3	665 595 680	691 688 857	699 347 941	695 870 751	697 620 096	698 196 101
R3L4	662 251 432	681 343 018	683 500 409	683 500 409	682 766 750	682 999 085
R4L1	723 276 168	754 707 390	764 266 997	764 266 997	762 687 598	763 002 450
R4L2	724 254 447	754 453 547	761 855 406	761 855 406	760 035 236	760 596 205
R4L3	722 434 044	751 351 849	755 869 038	755 869 038	754 672 008	754 766 718
R4L4	720 103 154	738 792 466	742 256 165	742 256 165	741 051 497	741 154 025

Table 1 Total travel times for the TimPassLib instances. Some instance names have been shortened for readability. We feature the best known lower bound and best known solution as reported on `https://timpasslib.aalto.fi/` as of July 4, 2025. Any solution by the integrated modulo network simplex, restricted integrated modulo network simplex, coarse or fine integrated tropical neighborhood search beating the currently best known solution is underlined. The blue solutions highlight which paradigm – MNS or TNS – is better.

instance	coarse ITNS time [s]	fine ITNS time [s]	coarse ITNS improvement	fine ITNS improvement	fine ITNS improving/total pairs
Erding20	201.671	599.255	3686	3174	2/80
Erding21	172.526	410.702	22991	20912	1/120
Hamburg	0.955	3.966	201416	201416	0/20
Schweiz	3600	3600	411681	307575	1/96
toy	0.224	1.323	8300	8300	1/80
toy2	0.260	2.798	0	0	0/10
regional	2.743	5.020	0	0	0/10
grid	13.171	13.162	126	114	0/60
long	105.494	183.894	3321	3321	0/30
metro	3.489	11.497	0	0	0/10
Stuttgart	—	—	0	0	0/0
R1L1	—	—	1850443	1380148	3/120
R1L2	—	—	1866879	1503935	6/155
R1L3	—	—	1448324	796428	4/78
R1L4	—	—	1180361	997131	2/84
R2L1	—	—	2910082	2364978	3/155
R2L2	—	—	3423781	2305179	2/140
R2L3	—	—	2587672	1898453	4/102
R2L4	—	—	976955	851117	3/50
R3L1	—	—	1868292	1173696	4/67
R3L2	—	—	1733097	924740	2/62
R3L3	—	—	1727845	1151840	5/51
R3L4	—	—	733659	501324	2/22
R4L1	—	—	1579399	1264547	2/50
R4L2	—	—	1820170	1259201	1/50
R4L3	—	—	1197030	1102320	2/40
R4L4	—	—	1204668	1102140	5/50

Table 2 Running times in seconds, improvement in terms of total travel time in comparison to the initial solution for the two ITNS algorithms, and improving/total number of OD pairs considered in Step (2) of Algorithm 16. The Stuttgart and RxLy instances hit the time limit of one hour.

instance	TimPassLib best known bound	TimPassLib previous best solution	parallelized RIMNS solution	previous gap [%]	improved gap [%]	relative improvement [%]	time [s]	parallelized coarse ITNS solution	parallelized coarse ITNS time [s]
Hamburg	139 892 927	141 629 305	141 610 697	1.24	1.23	1.07	2	141 842 262	1
Schweiz	60 084 289	62 622 935	62 484 232	4.23	3.99	5.46	25 474	64 906 482	606
toy	21 466	21 466	21 466	0.00	—	—	<1	21 466	<1
toy2	19 114	19 114	19 114	0.00	—	—	<1	19 198	<1
regional	1 804 642	1 834 884	1 827 124	1.68	1.25	25.66	25	1 855 092	1
grid	47 824	49 279	48 894	3.04	2.24	26.46	15	49 775	3
long	64 906 980	67 480 984	67 189 746	3.97	3.52	11.41	—	68 296 696	38
metro	11 978 129	12 019 079	11 997 040	0.34	0.16	53.82	200	12 029 681	2
Erding20	12 206 083	12 258 126	12 239 162	0.43	0.27	36.44	421	12 270 189	202
Erding21	12 206 083	12 307 765	12 258 531	0.83	0.43	48.42	410	12 298 178	411
Stuttgart	45 072 189 100	48 325 440 573	48 724 661 870	7.22	—	—	—	48 724 661 870	—
R1L1	522 575 407	542 908 145	540 895 611	3.89	3.51	9.90	34 262	546 503 687	18 489
R1L2	522 212 362	542 381 697	540 026 016	3.86	3.41	11.68	51 367	544 594 207	19 547
R1L3	522 199 838	543 067 240	540 484 607	4.00	3.50	12.38	59 193	544 803 285	16 801
R1L4	520 799 059	537 879 494	533 350 776	3.28	2.41	26.52	—	538 789 771	30 126
R2L1	650 575 045	681 061 389	674 214 817	4.69	3.63	22.46	—	682 608 885	29 379
R2L2	650 293 220	676 836 085	674 539 523	4.08	3.73	8.61	—	680 897 566	38 062
R2L3	649 767 761	675 793 893	672 506 998	4.01	3.50	12.51	—	677 313 411	31 854
R2L4	647 184 195	667 537 183	661 262 179	3.14	2.18	30.57	—	667 277 862	—
R3L1	665 804 283	694 086 648	689 439 206	4.25	3.55	16.47	—	697 177 072	53 182
R3L2	665 719 574	694 334 373	690 122 292	4.30	3.67	14.65	—	695 894 374	69 366
R3L3	665 595 680	691 688 857	686 486 157	3.92	3.14	19.90	—	693 936 887	57 760
R3L4	662 251 432	681 343 018	676 201 297	2.88	2.11	26.74	—	679 596 915	—
R4L1	723 276 168	754 707 390	751 888 172	4.35	3.96	8.97	—	758 684 327	510 03
R4L2	724 254 447	754 453 547	750 917 302	4.17	3.68	11.75	—	757 584 922	47 399
R4L3	722 434 044	751 351 849	747 309 217	4.00	3.44	14.00	—	751 346 581	—
R4L4	720 103 154	738 792 466	735 412 808	2.60	2.13	18.08	—	738 168 298	—

Table 3 We provide improved incumbents (underlined) for all but three TimPassLib instances after allotting more computing resources in the form of 96 CPU threads and a runtime limit of 24 hours wall time (time limit hit: —). Note that the toy instances have already been solved to optimality previously and that Stuttgart is notoriously challenging. The relative improvement is calculated by (old gap − new gap)/old gap.