Incremental Heuristics For Periodic Timetabling

Niels Lindner[1,2] [0000−0002−8337−4387] and
Christian Liebchen[3] [0000−0002−4311−2024]

[1] Freie Universität Berlin, Institut für Mathematik, Arnimallee 3, 14915 Berlin
[2] Zuse Institute Berlin, Department Network Optimization, Takustr. 7, 14915 Berlin
lindner@zib.de
[3] Technical University of Applied Sciences Wildau, Hochschulring 1, 15745 Wildau
liebchen@th-wildau.de

Abstract. We present incremental heuristics for the Periodic Event Scheduling Problem (PESP), the standard mathematical tool to optimize periodic timetables in public transport. The core of our method is to solve successively larger subinstances making use of previously found solutions. Introducing the technical notion of free stratifications, we formulate a general scheme for incremental heuristics for PESP. More practically, we use line and station information to create heuristics that add lines or stations one by one, and we evaluate these heuristics on instances of the benchmarking library PESPlib. This approach is indeed viable, and leads to new incumbent solutions for six PESPlib instances.

Keywords: Timetabling · Mixed-Integer Programming · Public Transport.

1 Introduction

Timetabling is an indispensable planning problem in public transport. Designing timetables carefully is not only vital for the attractiveness of a public transport system, but also for its operational and economic efficiency, since cost-sensitive tasks such as vehicle and crew scheduling build upon a timetable. As a large quantity of public transport networks is operated with a periodic pattern, in particular in Central Europe, there is hence a demand for periodic timetable optimization.

The Periodic Event Scheduling Problem (PESP) [11] is a mathematical framework that captures many aspects of periodic timetabling in public transport. Although there is a vast supply of primal heuristics (e.g., [1,2,8,9]), PESP is an NP-hard optimization problem that is also difficult to solve in practice: All instances of the benchmark library PESPlib [3] have withstood all attempts to solve them to proven optimality since the establishment of PESPlib in 2012.

Recently, the TimPassLib set has been published [10], and it contains background information on the 16 railway timetabling instances of PESPlib: The events that are to be scheduled are associated to *stations* and to *lines*.

We therefore suggest two incremental heuristics for periodic timetabling instances that share their structural properties with the PESPlib railway instances:

At first, we decompose the instance into a chain of subinstances, starting with the restriction to the busiest line, and adding less busier lines and the induced transfer activities successively. Secondly, we build another chain of subinstances, starting with the busiest station and the lines it is connected to, and then successively adding more stations. Both chains have the property that – due to a lack of headway constraints – a feasible solution of a subinstance can be extended to a feasible solution of the subsequent subinstance. We exploit this to obtain a conceptually simple heuristic, and demonstrate that it is in fact competitive.

We summarize the theoretical background in Section 2, which will allow us to describe our incremental heuristics formally in Section 3. The computational set-up and our results are presented in Section 4.

2 Theoretical Background

We briefly recall the Periodic Event Scheduling Problem in Section 2.1 before describing the structural requirements for our heuristic in Section 2.2.

2.1 The Periodic Event Scheduling Problem

An instance of the Periodic Event Scheduling Problem (PESP) [11] is a tuple (G, T, ℓ, u, w) that consists of an *event-activity network* (= digraph) $G = (V, A)$, a *period time* $T \in \mathbb{N}$, *lower bounds* $\ell \in \mathbb{Z}^A$ and *upper bounds* $u \in \mathbb{Z}^A$ on activity durations, and activity *weights* $w \in \mathbb{R}^A_{\geq 0}$. A *periodic timetable* is a vector $\pi \in \{0, 1, \ldots, T-1\}^V$ such that there is a *periodic tension* $x \in \mathbb{Z}^A$ satisfying

$$\forall a = (i, j) \in A: \quad \ell_a \leq x_a \leq u_a \quad \text{and} \quad \pi_j - \pi_i \equiv x_a \bmod T. \tag{1}$$

Given an instance (G, T, ℓ, u, w), the aim of PESP is to find a periodic timetable π with a compatible periodic tension x such that the weighted periodic tension $w^\top x$, or equivalently, the *weighted periodic slack* $w^\top(x - \ell)$, is minimized.

2.2 Free Stratifications

Let $I = (G, T, \ell, u, w)$ be a feasible PESP instance. By a subinstance I_k of I we mean the restriction of I to a subgraph $G_k = (V_k, A_k)$ of $G = (V, A)$.

Definition 1. *A* free stratification *of I is a sequence $(I_1, \ldots, I_n)$ of subinstances of I such that*

(1) *$I_n = I$,*
(2) *for all $k \in \{2, \ldots, n\}$, G_{k-1} is a subgraph of G_k,*
(3) *for all $k \in \{1, \ldots, n\}$, all arcs $a \in A_k \setminus A_{k-1}$ with at least one endpoint in V_{k-1} are* free*, i.e., $u_a - \ell_a \geq T - 1$.*

While (1) and (2) mean that we decompose I into a chain of subinstances, (3) implies the following theorem, whose proof we omit due to length restrictions.

Theorem 1. *Let $(I_1, \ldots, I_n)$ be a free stratification of a feasible PESP instance I, and let $k \in \{2, \ldots, n\}$. If π^{k-1} is a periodic timetable for I_{k-1}, then there is a periodic timetable π^k for I_k such that $\pi^k_i = \pi^{k-1}_i$ for all $i \in V_{k-1}$.*

3 Incremental Heuristics

We are now ready to formulate our incremental periodic timetabling heuristic, describing first a general scheme in Section 3.1. We then provide two illustrative incarnations, based on lines (Section 3.2) and stations (Section 3.3).

3.1 General Scheme

Consider a feasible PESP instance $I = (G, T, \ell, u, w)$ with a free stratification $(I_1, \dots, I_n)$. Our baseline algorithm is given in Algorithm 1.

Algorithm 1: Incremental heuristic for PESP with free stratification

Input: feasible PESP instance I with free stratification $(I_1, \dots I_n)$
Output: periodic timetable π on I

$\pi^0_{\text{full}} \leftarrow \emptyset$
$V_0 \leftarrow \emptyset$
for $k \leftarrow 1$ **to** n **do**
  $\tilde{\pi}^k \leftarrow$ periodic timetable on subinstance $\tilde{I}_k$ on $G_k[V_k \setminus V_{k-1}]$
  $\pi^k_{\text{initial}} \leftarrow \texttt{union}(\pi^{k-1}_{\text{full}}, \tilde{\pi}^k)$
  $\pi^k_{\text{fix}} \leftarrow \texttt{fix\_opt}(\pi^{k-1}_{\text{full}}, \pi^k_{\text{initial}})$
  $\pi^k_{\text{full}} \leftarrow \texttt{full\_opt}(\pi^k_{\text{fix}})$
end
return π^n_{full}

We iterate over the subinstances $I_1, \dots, I_n$. In each iteration k, we determine some periodic timetable $\tilde{\pi}^k$ on the subinstance $\tilde{I}_k$ on the subgraph $G_k[V_k \setminus V_{k-1}]$. By means of `union`, we then extend $\tilde{\pi}^k$ with a previously found timetable π^{k-1}_{full} on I_{k-1} to obtain a timetable π^k_{initial} on I_k. The existence is guaranteed by Theorem 1. The function `fix_opt` attempts to solve the PESP instance I_k with the additional requirement that the timetable on I_{k-1} is fixed to π^{k-1}_{full}, and we take π^k_{initial} as an initial solution. The output π^k_{fix} of `fix_opt` is then used an initial solution to an unrestricted optimization `full_opt` of I_k, whose output is π^k_{full}. Finally, the procedure returns π^n_{full}.

Algorithm 1 leaves several degrees of freedom, e.g., choosing the timetables $\tilde{\pi}^k$, and the details of the optimization processes behind `fix_opt` and `full_opt`. The main ingredient is of course a free stratification. When its subinstances are rather small, it is to be expected that determining $\tilde{\pi}^k$ and the `fix_opt` step are computationally always feasible, and also `full_opt` is tractable for small k.

Before confirming this intuition by computational experiments in Section 4, we will formulate two hands-on applications of Algorithm 1.

3.2 Incrementing Lines

In order to exploit line information, we make the following two assumptions:

(L1) There is a set $\mathcal{L}$ of *lines* and a map $L : V \to \mathcal{L}$.
(L2) Each activity $a = (i, j) \in A$ with $L(i) \neq L(j)$ is free.

These two assumptions are restrictive in the sense that they exclude, e.g., the typical modeling of headway activities (see, e.g., [5]). However, the 16 PESPlib railway instances `RxLy` for $x, y \in \{1, 2, 3, 4\}$ do satisfy (L1) and (L2) after slight preprocessing: Although there are a few headway arcs [7, §5.1], these turn out to be bridges in the event-activity network, which can be deleted [1, §3.2].

Suppose that (L1) and (L2) are satisified. Then any ordering $(\ell_1, \dots, \ell_n)$ of the elements of $\mathcal{L}$ yields a free stratification via $G_k := G[L^{-1}(\{l_1, \dots, l_k\})]$, i.e., the k-th subinstance is given by the events associated to the first k lines and the activities between them. A somehow related approach has been used in [9].

We suggest the following to sort the lines: Start with the line l_1 with largest *weighted span* $\sum_{a \in A[L^{-1}(l_1)]} w_a(u_a - \ell_a)$. When $k-1$ lines have been arranged, the next line l_k is one that intersects at least one of $l_1, \dots, l_{k-1}$, i.e., there is a *transfer* activity $a = (i, j) \in A$ with $L(i) = l_r$ for some $r \leq k - 1$ and $L(j) = l_k$ or vice versa, and maximizes the sum of weighted span of the activities in $A[L^{-1}(l_k)]$ and of all transfer activities between l_k and the previous lines $l_1, \dots, l_{k-1}$. We then continue this process. The idea is that the weighted span combines passenger usage in terms of w and optimization potential in terms of the span $u - \ell$, so that "important" lines come first. Since smaller instances are more likely to be solved optimally, in particular for `full_opt` it appears to be promising to reserve this privilege of the first rounds to the most important lines.

3.3 Incrementing Stations

We also propose to use stations in addition to line information satisfying (L1) and (L2). We hence further assume that there

(S1) There is a set $\mathcal{S}$ of *stations* and a map $S : V \to \mathcal{S}$.
(S2) Each activity $a = (i, j) \in A$ with $S(i) = S(j)$ is free or satisfies $L(i) = L(j)$.

If $(s_1, \dots, s_n)$ is any ordering of $\mathcal{S}$, then we obtain a free stratification by

$$G_k := G[S^{-1}(\{s_1, \dots, s_k\}) \cup L^{-1}(L(S^{-1}(\{s_1, \dots, s_k\})))],$$

i.e., the k-th subinstance is given by the events belonging to the first k stations and all events of lines that visit the first k stations, and all activities in between. At each step, we hence add arcs within a station, which are free by (S2), or we add a whole line, which is connected to the instance of the previous step by exclusively free arcs due to (L2).

We will later sort the stations $s \in \mathcal{S}$ in descending order with respect to the weighted span, i.e., $\sum_{a \in A[S^{-1}(s)]} w_a(u_a - \ell_a)$, again with the intuition that "busiest" stations are first.

3.4 Larger Bunches

The free stratifications in Section 3.2 and Section 3.3 add lines or stations one by one, thus creating a high number n of subinstances. We therefore suggest to coarsen the stratification by considering $(I_b, I_{2b}, I_{3b}, \dots, I_n)$ for a bunch size b.

instances	`R`x`L`y for $x, y \in \{1, 2, 3, 4\}$ [3,10]
free stratification	lines (as in Section 3.2), or stations (as in Section 3.3)
bunch size b	1, 2, …, 12 (cf. Section 3.4)
finding $\tilde{\pi}^k$	always trivial ($\tilde{I}_k$ with free arcs removed contains no cycles)
`fix_opt`	`ConcurrentPESP`, wall time limit: 10 min per iteration
`full_opt`	skip ($\pi^k_{\text{full}} := \pi^k_{\text{fix}}$), or `ConcurrentPESP`, wall time limit: 1 min per iteration

Table 1. Overview of computational experiments

4 Computational Experiments

We evaluate the incremental heuristic outlined in Section 3 on the 16 PESPlib railway instances. Data on lines and stations is provided by the TimPassLib [10]. After preprocessing as discussed in Section 3.2, the instances conform to the requirements (L1), (L2), (S1), (S2). We conduct a series of experiments whose parameters are summarized in Table 1. The `fix_opt` and `full_opt` procedures are powered by the solver `ConcurrentPESP` [1], including tropical neighborhood search [2] and split user cuts [6] with Gurobi 10 [4] as MIP solver.

Table 2 gives an overview of the results. It turns out that the incremental heuristics, especially the line-based one, is very competitive. We are able to produce new incumbent solutions for six PESPlib instances. This is particularly surprising, as we spend only 1 min on optimizing each instance via `full_opt`. Moreover, skipping `full_opt` and hence using only `fix_opt` for optimization is conceptually even simpler heuristic and still provides decent solutions. Concerning `fix_opt`, the optimality gap has been positive in only less than 5% of all runs, and it was always $\leq 0.6\%$ after 10 minutes. We finally note that higher bunch sizes are often, but not always advantageous.

References

1. Borndörfer, R., Lindner, N., Roth, S.: A concurrent approach to the periodic event scheduling problem. Journal of Rail Transport Planning & Management **15**, 100175 (2020). https://doi.org/10.1016/j.jrtpm.2019.100175
2. Bortoletto, E., Lindner, N., Masing, B.: Tropical Neighbourhood Search: A New Heuristic for Periodic Timetabling. In: D'Emidio, M., Lindner, N. (eds.) 22nd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2022). Open Access Series in Informatics (OASIcs), vol. 106, pp. 3:1–3:19 (2022). https://doi.org/10.4230/OASIcs.ATMOS.2022.3
3. Goerigk, M.: PESPlib – A benchmark library for periodic event scheduling (2022), `http://num.math.uni-goettingen.de/~m.goerigk/pesplib/`
4. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), `https://www.gurobi.com`
5. Liebchen, C., Möhring, R.H.: The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables — and Beyond. In: Geraets, F., Kroon, L., Schoebel,

Instance	with `full_opt`				without `full_opt`			
	Best obj.	Strat.	b	Time	Best obj.	Strat.	b	Time
R1L1	30 501 364	stations	2	2 528	32 158 122	lines	10	1 863
R1L2	31 165 708	lines	12	1 724	32 932 287	lines	10	1 402
R1L3	31 125 965	stations	1	5 183	32 645 175	lines	9	845
R1L4	27 693 907	stations	2	2 700	28 899 146	lines	8	817
R2L1	**41 730 227**	lines	2	2 079	44 051 010	lines	10	1 522
R2L2	41 293 379	lines	10	2 405	41 855 493	lines	12	2 174
R2L3	**37 952 185**	lines	10	1 129	39 540 917	lines	10	1 095
R2L4	**32 307 020**	lines	3	3 085	34 535 758	lines	11	868
R3L1	45 237 157	lines	10	1 947	46 190 273	lines	10	1 917
R3L2	45 812 583	lines	8	1 056	48 096 742	lines	8	535
R3L3	**40 424 380**	lines	1	6 736	43 764 798	lines	8	1 196
R3L4	33 542 154	lines	12	3 490	34 598 375	lines	12	2 936
R4L1	**48 939 815**	lines	10	2 372	51 594 813	lines	10	1 966
R4L2	49 139 234	lines	1	5 955	51 872 348	lines	12	2 850
R4L3	**45 177 738**	lines	10	2 806	47 207 825	lines	10	2 434
R4L4	38 382 967	lines	1	11 758	40 353 821	lines	10	2 316

Table 2. Overview of the results. The columns indicate the instance, the best found objective value in terms of weighted slack $w^\top(u-\ell)$, the stratification strategy, bunch size b, the wall time in seconds, and whether `full_opt` has been invoked. Objective values that are better than the current PESPlib incumbent are highlighted **bold**.

A., Wagner, D., Zaroliagis, C.D. (eds.) Algorithmic Methods for Railway Optimization. pp. 3–40. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74247-0_1
6. Lindner, N., Masing, B.: On the Split Closure of the Periodic Timetabling Polytope (2023), `http://arxiv.org/abs/2306.02746`
7. Masing, B., Lindner, N., Ebert, P.: Forward and Line-Based Cycle Bases for Periodic Timetabling. Operations Research Forum **4**(3), 53 (2023). https://doi.org/10.1007/s43069-023-00229-0
8. Nachtigall, K., Opitz, J.: Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations. In: Fischetti, M., Widmayer, P. (eds.) 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08). OpenAccess Series in Informatics (OASIcs), vol. 9 (2008). https://doi.org/10.4230/OASIcs.ATMOS.2008.1588
9. Pätzold, J., Schöbel, A.: A Matching Approach for Periodic Timetabling. In: Goerigk, M., Werneck, R. (eds.) 16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016). OpenAccess Series in Informatics (OASIcs), vol. 54, pp. 1:1–1:15 (2016). https://doi.org/10.4230/OASIcs.ATMOS.2016.1
10. Schiewe, P., Goerigk, M., Lindner, N.: Introducing TimPassLib – A library for integrated periodic timetabling and passenger routing. ZIB-Report 23-06, Zuse Institute Berlin (2023), `https://nbn-resolving.org/urn:nbn:de:0297-zib-89741`
11. Serafini, P., Ukovich, W.: A Mathematical Model for Periodic Scheduling Problems. SIAM Journal on Discrete Mathematics **2**(4), 550–581 (1989). https://doi.org/10.1137/0402049