




PATRICIA EBERT, BERENIKE MASING<sup>1</sup>, NIELS LINDNER<sup>2</sup>,  
AMBROS GLEIXNER<sup>3</sup>

## Sorting Criteria for Line-based Periodic Timetabling Heuristics

---

<sup>1</sup>  0000-0001-7201-2412

<sup>2</sup>  0000-0002-8337-4387

<sup>3</sup>  0000-0003-0391-5903

Zuse Institute Berlin  
Takustr. 7  
14195 Berlin  
Germany

Telephone: +49 30 84185-0  
Telefax: +49 30 84185-125

E-mail: [bibliothek@zib.de](mailto:bibliothek@zib.de)  
URL: <http://www.zib.de>

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

# Sorting Criteria for Line-based Periodic Timetabling Heuristics

Patricia Ebert<sup>1,2</sup>, Berenike Masing<sup>1</sup>[0000–0001–7201–2412],  
Niels Lindner<sup>1</sup>[0000–0002–8337–4387], and  
Ambros Gleixner<sup>1,2</sup>[0000–0003–0391–5903]

<sup>1</sup> Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany

<sup>2</sup> Hochschule für Technik und Wirtschaft Berlin, 10313 Berlin, Germany  
{ebert,masing,lindner,gleixner}@zib.de

**Abstract.** It is well-known that optimal solutions are notoriously hard to find for the Periodic Event Scheduling Problem (PESP), which is the standard mathematical formulation to optimize periodic timetables in public transport. We consider a class of incremental heuristics that have been demonstrated to be effective by Lindner and Liebchen (2023), however, for only one fixed sorting strategy of lines along which a solution is constructed. Thus, in this paper, we examine a variety of sortings based on the number, weight, weighted span, and lower bound of arcs, and test for each setting various combinations of the driving, dwelling, and transfer arcs of lines. Additionally, we assess the impact on the incremental extension of the event-activity network by minimizing resp. maximizing a connectivity measure between subsets of lines. We compare our 27 sortings on the railway instances of the benchmarking library PESPlib within the ConcurrentPESP solver framework. We are able to find five new incumbent solutions, resulting in improvements of up to 2%.

**Keywords:** Public Transport, Timetabling, Periodic Event Scheduling

## 1 Introduction

As many European countries use periodic timetables for public transport, their construction becomes an indispensable task. Even more, their optimization offers an inexpensive possibility to exploit the existing network in an efficient way.

A frequently used mathematical foundation allowing for the construction and optimization of periodic timetables is the Periodic Event Scheduling Problem (PESP) by Serafini and Ukovich [7], which is known to be NP-hard [5]. Although PESP can be formulated as a mixed-integer linear program, its instances are notoriously hard to solve not only in theory but also in practice, as is evidenced, e.g., by the fact that none of the instances of the benchmark library PESPlib [2] – introduced in 2012 – could be solved to proven optimality so far.

Besides efforts to strengthen dual bounds [4], Lindner and Liebchen showed with their primal incremental heuristic that improvements of the primal bound for PESPlib instances are still achievable [3]. They tested two approaches to incrementally build up and optimize subinstances by utilizing information about

the underlying structure, which became available through the recently published library TimPassLib [6]. Their approach consists of iteratively adding arcs to the instance based on line association and station association, respectively. In their computational experiment, they observed a superior behavior of the line-based approach, which quickly generated qualitatively competitive solutions for the PESPlib rail instances. However, the order in which the lines are processed has been fixed. Therefore, we want to examine in this work how different sortings of the lines affect the outcome of the incremental heuristic. To this end, we discuss nine sorting strategies that can be arbitrarily combined with three connectivity rules, and compare them by their best primal objective value with the computational setup of [3], using the ConcurrentPESP solver [1] as subroutine.

We define the Periodic Event Scheduling Problem and explain a line-based incarnation of the incremental heuristic in Section 2. After discussing our sorting strategies in Section 3, we conclude with computational experiments in Section 4.

## 2 Problem Definition and Incremental Heuristic

We begin with some necessary theoretical background.

### 2.1 Periodic Event Scheduling Problem

For the Periodic Event Scheduling Problem (PESP) [7] we are given a *period time*  $T \in \mathbb{N}$  and an *event-activity-network*  $\mathcal{N} = (V, A)$  with *lower bounds*  $l \in \mathbb{Z}^A$  and *upper bounds*  $u \in \mathbb{Z}^A$  on the activity durations as well as activity *weights*  $w \in \mathbb{R}_{\geq 0}^A$ . A *periodic timetable*  $\pi \in \{0, 1, \dots, T-1\}^V$  is a periodic node potential such that for each activity  $a = (i, j) \in A$  the periodic span constraint

$$l_a \leq (\pi_j - \pi_i) \bmod T =: x_a \leq u_a$$

is satisfied. Given an instance  $(\mathcal{N}, T, l, u, w)$  the optimization version of PESP asks for a feasible periodic timetable  $\pi$  minimizing the weighted periodic tension  $w^\top x$ . We call an activity  $a = (i, j) \in A$  *free* if  $u_a - l_a \geq T - 1$  holds for its span, as then any choice of  $\pi_i$  and  $\pi_j$  corresponds to a valid tension  $x_a$ .

### 2.2 Free Stratifications

Let  $I = (\mathcal{N}, T, l, u, w)$  be a feasible PESP instance and  $I_k$  a subinstance of  $I$  where we restrict  $\mathcal{N} = (V, A)$  to a subgraph  $\mathcal{N}_k = (V_k, A_k)$ . A *free stratification* [3] of  $I$  is a sequence  $(I_1, \dots, I_n)$  of subinstances of  $I$  such that

1.  $I = I_n$ ,
2. for all  $k \in \{2, \dots, n\}$ ,  $\mathcal{N}_{k-1}$  is a subgraph of  $\mathcal{N}_k$ ,
3. for all  $k \in \{2, \dots, n\}$ , all arcs  $a \in A_k \setminus A_{k-1}$  with at least one endpoint in  $V_{k-1}$  are free.

This allowed Lindner and Liebchen [3] to state an extension theorem:

**Theorem 1 ([3, Thm. 1]).** *Let  $(I_1, \dots, I_n)$  be a free stratification of a feasible PESP instance  $I$ , and let  $k \in \{2, \dots, n\}$ . If  $\pi^{k-1}$  is a periodic timetable for  $I_{k-1}$ , then there is a periodic timetable  $\pi^k$  for  $I_k$  such that  $\pi_i^k = \pi_i^{k-1}$  for all  $i \in V_{k-1}$ .*

### 2.3 Incremental Line-Based Heuristics for PESLib Instances

We will consider a special case of free stratifications, as the TimPassLib data [6] endows the PESLib rail instances with a special structure: There is a set  $\mathcal{L}$  of *lines* and a map  $L : V \rightarrow \mathcal{L}$  that associates each event to a line with the property that each activity  $a = (i, j) \in A$  with  $L(i) \neq L(j)$  is free. Each such activity is a *transfer* activity, while activities with both events corresponding to the same line are either *driving* or *dwelling* activities. A sorting  $(\ell_1, \dots, \ell_n)$  of the lines in  $\mathcal{L}$  induces a free stratification by setting  $I_k$  to be the subinstance given by the events of the first  $k$  lines and all activities between them [3]. Another specialty of the PESLib instances is that the subgraph  $G_k[V_k \setminus V_{k-1}]$ , i.e., the activities that belong to line  $\ell_k$ , is always a disjoint union of two paths.

---

**Algorithm 1:** Line-based Incremental Heuristic
 

---

**Input:** PESLib rail instance  $I$  with line association map  $L : V \rightarrow \mathcal{L}$   
**Output:** periodic timetable  $\pi$  on  $I$

- 1  $\pi_{\text{full}}^0 \leftarrow \emptyset, V_0 \leftarrow \emptyset, A_0 \leftarrow \emptyset$
- 2  $(\ell_1, \dots, \ell_n) \leftarrow \text{sorting of } \mathcal{L}$
- 3 **for**  $k \leftarrow 1$  **to**  $n$  **do**
- 4      $F_k \leftarrow \text{spanning forest on } I_k \text{ s.t. all non-forest arcs are in } A_{k-1} \text{ or free}$
- 5      $\pi_{\text{initial}}^k \leftarrow \text{extend}(\pi_{\text{full}}^{k-1}, F_k)$
- 6      $\pi_{\text{fix}}^k \leftarrow \text{fix\_opt}(\pi_{\text{full}}^{k-1}, \pi_{\text{initial}}^k)$
- 7      $\pi_{\text{full}}^k \leftarrow \text{full\_opt}(\pi_{\text{fix}}^k)$
- 8 **end**
- 9 **return**  $\pi_{\text{full}}^n$

---

This allows for the incremental Algorithm 1. In the  $k$ -th iteration, we are given a feasible timetable  $\pi_{\text{full}}^{k-1}$  on  $I_{k-1}$ . We construct a spanning forest  $F_k$  on  $I_k$  such that all non-forest arcs are part of  $I_{k-1}$  or free. This essentially means to take an arbitrary spanning forest on  $I_{k-1}$ , adding the two paths of line  $\ell_k$ , and taking at most two free arcs that connect line  $\ell_k$  with one of the previous lines. We then construct a timetable  $\pi_{\text{initial}}^k$  on  $I_k$  by traversing along  $F_k$ , using the tension of  $\pi_{\text{full}}^{k-1}$  on arcs of  $I_{k-1}$ , and lower bounds otherwise. This initial solution is then improved in two rounds. Firstly, in **fix\_opt** the subinstance  $I_k$  is optimized with  $\pi_{\text{initial}}^k$  as an initial solution and  $\pi_{\text{full}}^{k-1}$  still being fixed on  $I_k$ , resulting in the periodic timetable  $\pi_{\text{fix}}^k$ . Secondly, the fixation is dropped and  $I_k$  is optimized in **full\_opt** with  $\pi_{\text{fix}}^k$  as an initial solution. The resulting timetable is denoted by  $\pi_{\text{full}}^k$ . The procedure is an incarnation of the algorithm in [3].

## 3 Sortings of Lines

As one can observe that subinstances can only be solved to optimality in earlier iterations of the incremental heuristic due to their smaller size, we motivate adding lines with a high impact on the objective value early on. For each sorting, we define a measure that assigns a line a value of importance, which then allows us to sort the lines in descending order. These measures take into account the

number (**n**), the weights (**w**), the weighted span (**ws**), and the lower bound (**l**) of arcs, which is evaluated by differentiation of driving (**-d**), versus driving and dwelling arcs (**-dd**), all outgoing arcs of a line (**-o**), and transfer arcs (**-t**), respectively. Moreover, we present a scheme for minimizing and maximizing the connectivity of the added lines in the sortings.

From a passenger’s perspective, lines connecting many stations, preferably with numerous transfer possibilities, are desired. To model that, we count for the measure (**n-d**) the number of driving arcs for each line and for (**n-o**) the number of driving, dwelling, and outgoing transfer arcs, which is given by the total number of outgoing arcs for the nodes corresponding to the line.

Since PESP minimizes the weighted periodic tension, we start for a second class of measures with lines that imply higher weights. For PESPlib instances the weight of an activity corresponds to the passenger numbers. Thus, by simply summing the weights for all driving arcs of a line in measure (**w-d**), we cumulate the passenger numbers of each section of the line and with (**w-dd**) we also add dwelling arcs, thus also taking into account how long all passengers travel along this line. However, driving and dwelling arcs in the PESPlib rail instances possess in general a significantly smaller span  $u-l$  as transfer arcs. Since the span directly impacts the optimization potential, we incorporate the transfer activities for a line in measure (**w-o**) and (**w-t**). For measure (**w-o**) we accumulate the weights of all driving, dwelling, and outgoing transfer activities of the line and in measure (**w-t**) the weights of all transfer arcs regardless whether they are out- or ingoing.

Lindner and Liebchen consider the weighted span, as the weight  $w$  incorporates passenger usage and the span  $u-l$  the optimization potential [3]. We examine the weighted span for driving and dwelling arcs in (**ws-dd**) and for all outgoing arcs in (**ws-o**). Due to a lack of information on distances, speeds, and station positions for our instances, we also utilize (**l-dd**) as the sum of the lower bounds on the duration of driving and dwelling activities of a line. The idea is that long lines covering large areas of the network are included first.

We complete our experiment by minimizing resp. maximizing the connectivity of the subinstances. We start for (**min-**) resp. (**max-**) with the most important line according to one of the previous measures, and then add the lines one by one. In each iteration, we consider the set of all remaining lines with a minimal resp. a maximal number of connections to the subinstance in terms of transfer activities and add the line which maximizes the measure. The hope for (**min-**) is, that due to the lower connectivity, we are able to obtain optimal solutions for a larger number of subinstances. In contrast, (**max-**) allows for early optimal solutions of the more challenging highly connected parts of the network.

## 4 Computational Experiment

We tested our heuristic on each of the 16 railway instances of PESPlib for our various line-sortings on an Intel Xeon E3-1270 machine with 32 GB RAM. In each iteration, we allowed for 60 seconds wall time for the `full_opt` procedure in line 7 of Algorithm 1 in ConcurrentPESP [1], and impose no time limit on `fix_opt`

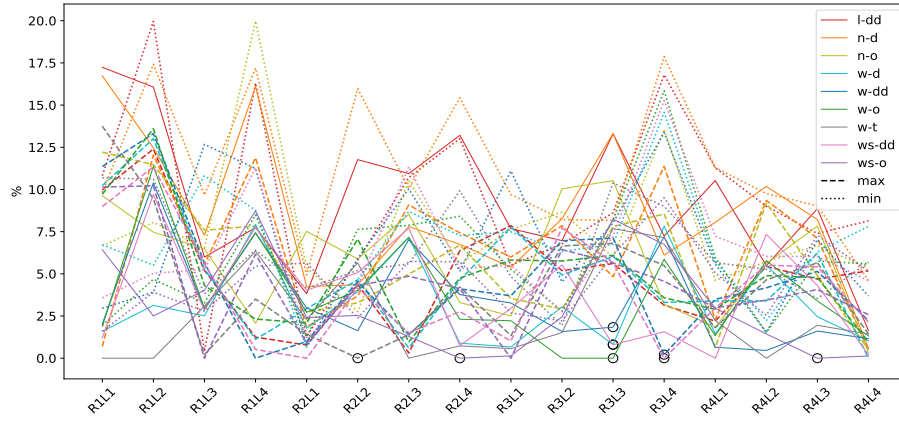


Fig. 1: Relative best gap of different line sortings: Highlighted values correspond to improvements w.r.t. current best PESLib incumbent

in line 6. We display our results in Figure 1 by the gap of the final primal value of each sorting in comparison to the best performing one for each instance. We observe that the (**min**)-runs (dotted lines) are never the best choice: They tend to be above the dashed and solid lines, so that sorting the lines by low connectivity and maintaining optimally solved subinstances longer does not pay off. What is further evident from Figure 1 is that – while there is no clear winner which consistently provides the best solution – the best performing sorting always involves the arc weights in some form: E.g., for R1L1 and R1L2 (**w-t**) is the best choice, while it is (**max-ws-o**) for R1L3. In contrast, any sorting based on lower bounds (**l-dd**) and number of arcs, (**n-d**), (**n-dt**), performs in general significantly worse. We thus conclude that weight-based sortings are preferable. This is further supported by Figure 2, where the accumulated relative gap over all 16 PESLib instances is depicted. This plot suggests that weights play a larger role than the influence of connectivity, as indicated by larger average gaps.

However, a close look at Figure 1 reveals that connectivity can, in fact, be helpful to consider, since – while not the best on average – **max**-sortings provide the best solutions for some instances. By Figure 1 it becomes evident that for each instance – with the exception of R4L1 – the best sorting involves weights (**w**), and considers either transfer arcs (which are included in (**-t**) and (**-o**)), or is a max-connectivity sorting. We conclude that sortings based on weighted connection arcs between lines are the preferred choice for our heuristic.

In the end, applying the heuristic by Lindner and Liebchen [3] with new sortings resulted in 5 new incumbent solutions for PESLib (see Table 1).

**Acknowledgments.** Berenike Masing: Research Campus MODAL, funded by the German Federal Ministry of Education and Research (BMBF) (fund no. 05M20ZBM).

**Disclosure of Interests.** The authors have no competing interests to declare.

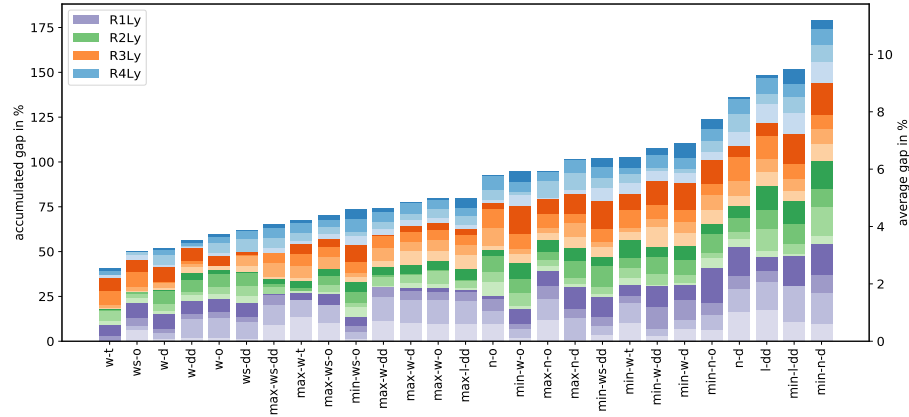


Fig. 2: Ranking of relative best gaps of different line sortings, color blocks correspond to one instance

instance	incumbent new	incumbent old	improvement	by sorting
R2L2	40424788	40642186	0.54%	(max-w-t)
R2L4	32286164	32307020	0.06%	(ws-o)
R3L3	39674349	40424380	1.89%	(ws-dd)
R3L4	32889035	33335852	1.36%	(max-ws-dd)
R4L3	45138727	45177738	0.09%	(ws-o)

Table 1: New incumbents in comparison to current best solutions

## 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. Goerigk, M.: PESPLib – A benchmark library for periodic event scheduling (2012), <http://num.math.uni-goettingen.de/~m.goerigk/pesplib/>
3. Lindner, N., Liebchen, C.: Incremental Heuristics for Periodic Timetabling. Tech. Rep. 23-22, Zuse-Institut Berlin (2023)
4. Masing, B., Lindner, N., Ebert, P.: Forward and Line-Based Cycle Bases for Periodic Timetabling. *Operations Research Forum* **4**(5) (2023). <https://doi.org/10.1007/s43069-023-00229-0>
5. Odijk, M.A.: Construction of periodic timetables, part 1: A cutting plane algorithm. Tech. Rep. 94-61, TU Delft (1994)
6. 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>
7. Serafini, P., Ukovich, W.: A Mathematical Model for Periodic Scheduling Problems. *SIDMA* **2**(4), 550–581 (1989). <https://doi.org/10.1137/0402049>