Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

STANLEY SCHADE, THOMAS SCHLECHTE, JAKOB WITZIG

Structure-Based Decomposition for Pattern-Detection for Railway Timetables

ZIB Report 17-40 (Juli 2017)

Zuse Institute Berlin
Takustr. 7
D-14195 Berlin

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

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

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

Structure-Based Decomposition for Pattern-Detection for Railway Timetables*

Stanley Schade[1], Thomas Schlechte[2], and Jakob Witzig[3]

[1] Zuse Institute Berlin, Mathematics of Transportation and Logistics,
Takustr. 7, 14195 Berlin, Germany, schade@zib.de

[2] LBW Optimization GmbH,
Obwaldener Zeile 19, 12205 Berlin, Germany, schlechte@lbw-optimization.de

[3] Zuse Institute Berlin, Mathematical Optimization Methods,
Takustr. 7, 14195 Berlin, Germany, witzig@zib.de

Abstract. We consider the problem of pattern detection in large scale railway timetables. This problem arises in rolling stock optimization planning in order to identify invariant sections of the timetable for which a cyclic rotation plan is adequate. We propose a dual reduction technique which leads to an decomposition and enumeration method. Computational results for real world instances demonstrate that the method is able to produce optimal solutions as fast as standard MIP solvers.

1 Introduction

The timetable is the starting point of a rotation planner. The objective is to assign job sequences to the available railway vehicles, such that every trip of the timetable is covered. In [4] the timetable is split up into distinct parts that each consist of repeating patterns during the planning process. This leads to the pattern detection problem, which was modeled using a mixed integer program. In this paper we investigate alternative solution strategies for this model in comparison to solving the model using a generic MIP solver.

In Section 2 we give an outline what the pattern detection problem is and how it arises. Subsequently, we present two greedy heuristics and a dual reduction that divides the problem into components that can be enumerated. In Section 4 we evaluate the run-time and accuracy of the presented algorithms.

2 Pattern-Detection

With regard to timetable patterns, it is only relevant whether two days of the timetable are equal with respect to the trips that are operated on these days.

*The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM).

Hence, a number can be assigned to each day, such that these numbers for two days are equal if and only if the trips to be operated agree. Note that the timetable has a weekly structure. Thus, generally two days are only compared if they correspond to the same weekday, e.g. two consecutive Mondays. As a result, we define a timetable to be a finite sequence of integers. The length of this finite sequence has to be a multiple of seven. Any subsequence of seven consecutive days of a timetable is a *pattern*. Because of the cyclic structure of patterns, we agree to start with the day that corresponds to Sunday, then Monday, Tuesday and so on if we write them down. An example of a timetable is $T = (1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2)$. Let us assume that the first day of T is a Sunday. Then, it contains the patterns $A = (1, 1, 1, 1, 1, 2, 2)$, $B = (2, 1, 1, 1, 1, 2, 2)$ and $C = (2, 1, 1, 1, 1, 1, 2)$. The pattern A matches the first 14 days of T, B matches days 8 to 20 and C matches the last seven days. If a pattern matches at least 8 consecutive days of a timetable, we say that it *covers* these days. Hence, A also covers the first 14 days of T and B also covers the days 8 to 20, but C does not cover any part of T. A more formal definition of the cover relation can be found in [4], but is left out here due to space constraints. More information on cyclic rotation planning with a period of one week can be found in [1]. During the planning process, the timetable for a year is developed gradually to include more and more details. A part of the timetable that is covered by a pattern has a weekly periodic structure. Therefore, one can also use a rotation plan with such a structure for this part. We aim to identify a few patterns that cover as much of the timetable as possible. For each of these patterns a rotation plan needs to be developed. The pattern detection, thus, is a useful tool in early planning stages. Determining patterns that cover parts of a timetable can be done by simple linear preprocessing. In practice, one aims to select only a few relevant patterns that cover as much of the timetable as possible, since each pattern corresponds to a rotation plan that needs to be developed. Parts of the timetable usually also lack a periodic structure and are not covered by patterns, e.g., extended holiday periods like Christmas. The following mixed integer program to identify relevant patterns was presented in [4].

$$\min -\sum_{i=1}^{n} x_i + 8 \sum_{j}^{m} y_i \tag{1}$$

$$s.t. \quad x_i - \sum_{j\,:\, j \text{ covers } i} y_j \leq 0 \qquad \forall i = 1, \ldots, n \tag{2}$$

$$x_i \in [0, 1] \qquad \forall i = 1, \ldots, n$$

$$y_j \in \{0, 1\} \qquad \forall j = 1, \ldots, m$$

Let n be the number of days and m the number of patterns. Setting the binary variable y_j to 1 means that pattern j is selected. In an optimal solution a day i is covered if and only if $x_i = 1$. In this case the constraint (2) ensures the existence of a pattern j that covers i.

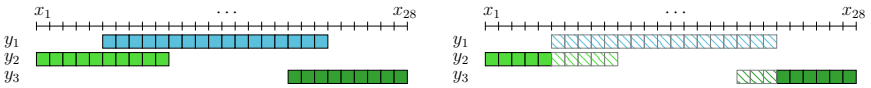

Fig. 1. Structure-based decomposition. Long pattern 1 covers 9 days exclusively (left). This pattern will be part of at least one optimal solution. The search space decomposes into two independent parts (right) after removing pattern 1, all days in $\mathcal{C}^{-1}(1)$, and all days covered by 1 that are in $\mathcal{C}^{-1}(2)$ and $\mathcal{C}^{-1}(3)$.

3 Structure-Based Propagation

In this section, we present two structure-based procedures for solving the pattern detection problem presented in the previous section. One procedure aims to decompose the search space into independent components, such that the resulting components are (hopefully) easier to solve. The other procedure is a greedy heuristic.

In the following, we denote set of patterns j *covering a day* i by $\mathcal{C}(i)$. Analogous, the set of days i that are *covered by a pattern* j is denoted by $\mathcal{C}^{-1}(j)$. Moreover, the set of days i that are covered by a *unique* pattern j is denoted by

$$\mathcal{U}(j) := \{i = 1, \ldots, n : \ j \in \mathcal{C}(i) \text{ and } |\mathcal{C}(i)| = 1\}.$$

We call a pattern j a *long pattern* if and only if $|\mathcal{U}(j)| \geq 9$.

Due to the fact that choosing a pattern that covers at most seven days will lead to a deterioration of the objective value, every y_j with $|\mathcal{C}^{-1}(j)| \leq 7$ can be fixed to 0. In fact, such patterns are ruled out by the preprocessing.

Decomposition by Days The special structure of the objective function of (1) allows us to determine patterns that will be part of at least one optimal solution. Selecting a long pattern always leads to an improvement of the objective function value by at least 1, because they cover at least 9 days that are covered by no other pattern. Usually, reductions guaranteeing that at least one optimal solution is preserved are called *dual reductions*, e.g., propagation with the objective function. On the other hand, a reduction that preserves all optimal solutions is called *primal*. In our case, we use a dual argument, i.e., the objective function, but we can guarantee that all optimal solution will be preserved.

Two patterns are called *overlapping* if and only if they mutually cover at least one day of a timetable. Consider a pattern j that has no overlap with any other pattern. Clearly, whether we set y_j to 0 or 1 does not influence the other patterns. If j overlaps with a pattern k, the objective function value may be improved by setting y_j or y_k to 1. But it can be possible that the objective value does not improve if both variables are set to 1 at the same time. Let us call two patterns j and k *connected* if they overlap or they are both connected to a third pattern l. In our decomposition approach we aim to split the search space into smaller pieces that are (hopefully) easier to solve, e.g., by complete enumeration or a MIP solver. To decompose an instance, we first remove all long patterns as

Algorithm 1 Structure-Based Propagation Procedure

$(x, y) \leftarrow (0, 0)$ ▷ initialize zero solution
$\mathcal{X} \leftarrow \{1, \dots, n\}$, $\mathcal{Y} \leftarrow \{1, \dots, m\}$ ▷ initialize index set of days and patterns
for all $j \in \{1, \dots, m\}$ with $|\mathcal{U}(j)| \geq 9$ **do** ▷ apply trivial fixings
  $y_j \leftarrow 1$; $\mathcal{Y} \leftarrow \mathcal{Y} \setminus j$
  for all $i \in \mathcal{C}^{-1}(j)$ **do**
    $x_i \leftarrow 1$; $\mathcal{X} \leftarrow \mathcal{X} \setminus i$
    for all $k \in \mathcal{Y}$ with $i \in \mathcal{C}^{-1}(k)$ **do**
      $\mathcal{C}^{-1}(k) \leftarrow \mathcal{C}^{-1}(k) \setminus i$
while $\mathcal{X} \neq \emptyset$ **do**
  $s \leftarrow \phi_{\mathcal{Y}}(\alpha, \beta)$ ▷ get current scores
  Get a permutation π such that $s_{\pi_j} \geq s_{\pi_{j+1}}$ for all $j \in \mathcal{Y}$
  $success \leftarrow false$
  for $j = 1, \dots, |\mathcal{Y}|$ **do** ▷ find pattern to fix with highest scores
    if $|\mathcal{C}^{-1}(\pi_j^{-1})| \geq 9$ **then**
      $y_{\pi_j^{-1}} \leftarrow 1$; $\mathcal{Y} \leftarrow \mathcal{Y} \setminus \pi_j^{-1}$; $success \leftarrow true$
      for all $i \in \mathcal{C}^{-1}(\pi_j^{-1})$ **do**
        $x_i \leftarrow 1$; $\mathcal{X} \leftarrow \mathcal{X} \setminus i$
        for all $k \in \mathcal{Y}$ with $i \in \mathcal{C}^{-1}(k)$ **do**
          $\mathcal{C}^{-1}(k) \leftarrow \mathcal{C}^{-1}(k) \setminus i$
      break
  if $!success$ **then** ▷ stop if no pattern was chosen
    break
return (x, y)

depicted in Figure 1. In the computational experiments, we used an enumeration approach and never had to enumerate more than 400 solutions with this strategy. But potentially, an instance could contain patterns that cover several different parts of the year and foil the decomposition. In this case using a MIP solver would be superior with regard to performance.

Heuristic A simpler approach is to score patterns and to greedily choose the patterns with the highest score one after another. We use the length of a pattern or the number of days that are uniquely covered as its score. To generalize this, we can use a scoring function $\phi_{\mathcal{Y}}(\alpha, \beta) = (\alpha \cdot |\mathcal{C}^{-1}(j)| + \beta \cdot |\mathcal{U}(j)|)_{j \in \mathcal{Y}}$. The heuristic looks for a pattern j with highest score, such that setting y_j to 1 leads to an improvement of the objective value. If no such pattern can be found, the heuristic terminates. Otherwise, the days covered by j are removed from the timetable. Thus, the scores of the patterns that overlap with j need to be recalculated. To avoid unnecessary update steps, it is reasonable to select all long patterns beforehand. The full heuristic is given as pseudocode in Algorithm 1.

Note that the heuristic may lead to suboptimal solutions. Say, we use the length of patterns as score, i.e., $\alpha = 1, \beta = 0$. We have three patterns 1, 2 and 3 of lengths 10, 11 and 10, respectively. They are arranged in a similar way as the patterns in Figure 1 with 1 and 2 having an overlap of 3 days and 2 and 3 having an overlap of 2 days. In this case the heuristic would first select 2, because it is

the longest pattern, and set y_2 to 1. However, it can easily be checked that we have $y_1 = y_3 = 1$ and $y_2 = 0$ for the optimal solution. A similar example can also be constructed for the case $\alpha = 0, \beta = 1$.

4 Computational Results

The decomposition procedure presented in Section 3 is used to decompose the problems into smaller pieces, which are solved by a complete enumeration afterwards. In the following we will refer to this by `Enumerate`. The heuristic (cf. Algorithm 1) runs with the scoring function $\phi_{\mathcal{Y}}$ and parameters $(1, 0)$ and $(0, 1)$.

A test set of 22 real-world instances provided by DB Fernverkehr AG is used. The number of patterns is shown in Table 1. All instances cover a time horizon of 364 days. All procedures were implemented in Python. The experiments were performed on a Dell Precision Tower 3620 with 3.50 GHz and 32 GB main memory.

In Table 1 we use `Enumerate` as a base line, for which we give the optimal objective value and running times in *ms*. For the heuristics, we instead give the optimality gap[4] and factors w.r.t. the base line. In [4] the arising MIP (1) was solved using the academic non-commercial mixed integer programming solver SCIP [3] and the according python interface PySCIPOpt [2]. However, all pattern detection problems as described in this article have a time horizon of one year and even for the largest instance the number of arising patterns cannot exceed 52. Such instances are not challenging for a sophisticated MIP solver and, therefore, we omit the SCIP running times in Table 1. Surprisingly, the heuristics determine the optimal solution in all cases, but one. However, as demonstrated in Section 3, examples where they do not find an optimal solution are easy to construct. In contrast to that `Enumerate` guarantees optimality and is still competitive with respect to the running time. `Enumerate` could even be further improved by solving the independent subproblems in parallel.

5 Conclusion

In this paper we presented an enumerative decomposition method and a heuristic for solving the pattern detection problem that arises in the context of railway rotation planning. It is a pity that the real-world instances are not challenging for a generic MIP solver. However, all presented methods perform quite good and it is funny that there is only one "bad" instance where the heuristic did not find an optimal solution.

Acknowledgments The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM).

[4] Gap to optimality: $|\text{primalbound} - \text{dualbound}| / \min\{|\text{primalbound}|, |\text{dualbound}|\}|$ if both bounds have same sign, or infinity, if they have opposite sign.

Table 1. Detailed computational results on 22 real-world instances.

Instance		`Enumerate`		$\phi_{\mathcal{Y}}(1,0)$		$\phi_{\mathcal{Y}}(0,1)$	
Name	m	ObjVal	Time	Gap	TimeQ	Gap	TimeQ
DB1	23	-201	0.57	0.00%	0.60	0.00%	0.59
DB2	20	-230	0.53	0.00%	0.49	0.00%	0.49
DB3	29	-187	0.69	0.00%	0.81	0.00%	0.81
DB4	26	-205	0.61	0.00%	0.66	0.00%	0.66
DB5	21	-311	1.27	0.00%	0.18	0.00%	0.17
DB6	25	-256	1.54	0.00%	0.19	0.00%	0.19
DB7	18	-295	0.94	0.00%	0.20	0.00%	0.19
DB8	9	-322	0.44	0.00%	0.36	0.00%	0.35
DB9	28	-116	0.72	0.00%	0.91	0.00%	0.91
DB10	18	-281	0.59	0.00%	0.35	0.00%	0.35
DB11	15	-301	0.64	0.00%	0.24	0.00%	0.24
DB12	11	-323	0.80	0.00%	0.15	0.00%	0.19
DB13	16	-312	0.52	0.00%	0.27	0.00%	0.26
DB14	23	-225	4.84	0.00%	0.07	*1.24%*	0.09
DB15	24	-117	0.86	0.00%	0.46	0.00%	0.47
DB16	11	-318	1.34	0.00%	0.16	0.00%	0.16
DB17	10	-330	0.48	0.00%	0.28	0.00%	0.28
DB18	21	-215	0.57	0.00%	0.45	0.00%	0.45
DB19	12	-329	0.77	0.00%	0.17	0.00%	0.17
DB20	10	-339	2.36	0.00%	0.07	0.00%	0.07
DB21	16	-250	0.49	0.00%	0.42	0.00%	0.41
DB22	20	-282	1.01	0.00%	0.23	0.00%	0.23

References

1. Borndörfer, R., Reuther, M., Schlechte, T., Waas, K., Weider, S.: Integrated Optimization of Rolling Stock Rotations for Intercity Railways. Transportation Science 50(3), 863–877 (2016)
2. Maher, S., Miltenberger, M., Pedroso, J.P., Rehfeldt, D., Schwarz, R., Serrano, F.: PySCIPOpt: Mathematical Programming in Python with the SCIP Optimization Suite. In: Mathematical Software – ICMS 2016. vol. 9725, pp. 301 – 307 (2016)
3. Maher, S.J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R.L., Hendel, G., Koch, T., Lübbecke, M.E., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 4.0. Tech. Rep. 17-12, ZIB, Takustr.7, 14195 Berlin (2017)
4. Schade, S., Borndörfer, R., Breuer, M., Grimm, B., Reuther, M., Schlechte, T., Siebeneicher, P.: Pattern Detection For Large-Scale Railway Timetables. In: Proceedings of the IAROR conference RailLille (2017)