Solving Public Railway Transport Networks with SAT

Peter Großmann\textsuperscript{1,2}, Steffen Hölldobler\textsuperscript{1}, Norbert Manthey\textsuperscript{1}, Karl Nachtigall\textsuperscript{2}, Jens Opitz\textsuperscript{2}, and Peter Steinke\textsuperscript{1}

\textsuperscript{1} Knowledge Representation and Reasoning Group, Faculty of Computer Science, TU Dresden
\{norbert,sh\}@iccl.tu-dresden.de, peter.steinke@tu-dresden.de
\textsuperscript{2} Chair of Traffic Flow Science, Faculty of Transport and Traffic Science, TU Dresden
peter.grossmann2@mailbox.tu-dresden.de, \{karl.nachtigall,jens.opitz\}@tu-dresden.de

Abstract. SAT solvers have been already successfully applied in several industrial fields, which are not directly related to propositional logic. In this work, the periodic event scheduling problem (PESP) will be presented and, furthermore, the order encoding from a PESP instance to a SAT instance. The \textit{NP}-complete PESP is particularly important in several traffic scenarios with periodic properties. Most native domain solvers cannot solve large instances within a given time frame. This will be omitted by a rather short time consuming conversion from PESP to SAT and a fast state-of-the-art SAT solver, in order to achieve a fast calculated solution of a PESP instance.

1 Introduction

Modelling and automatic solving of large periodic event networks, like a time table for the railway network of Germany, is still an open scientific field. The model, the so called periodic event scheduling problem (PESP), is well defined, but lacks in efficient runtime behaviour for all state-of-the-art PESP solvers, if the instances become very large. Since the PESP is a \textit{NP}-complete problem \cite{3}, there will always be instances, which cannot be solved in any reasonably given time frame, unless \(\mathcal{P} = \mathcal{NP}\). However, for a lot of \textit{NP}-hard problems, there exist efficient algorithms to solve very hard and huge instances in an acceptable solving time.

One example is the solving of propositional formulas – the so called SAT solving. Several industrial problems have already been reformulated as propositional formulas and been solved by SAT solvers. Hence, these are generic solvers to solve different domains.

This work tries to follow this approach: reformulating a model of the periodic event network domain to a propositional formula and solving this instance with a state-of-the-art SAT solver. The transition, or reduction, to this formula is called encoding. In this work two different encodings will be presented to reduce a PESP instance into a SAT instance.
The core idea of the order encoding, with respect to the PESP domain, is the utilization of the constraint’s definition, which is strictly linear – even the identity function. This information can be greatly used for a domain like the propositional formulas, which just have two values as domain of its variables.

Another effort in this work will be done by proofing the method to be sound and correct. Consequently, the use of this method is valid and can replace a traditional PESP solver.

2 Notations and Preliminaries

The following two sections give the required background on the satisfiability testing problem (SAT) and the periodic event scheduling problem (PESP). The notation that is used in this work is also specified.

2.1 Satisfiability Problem

The SAT problem is based on propositional logic. For each Boolean variable \( v \) there is a positive and a negative literal, denoted by \( v, \overline{v} \) respectively. A clause is a disjunction of literals, is represented as finite set of literals and will be denoted by \( C = [l_1, \ldots, l_n] \). A formula in conjunctive normal form (CNF) is a conjunction of clauses and can also be represented as finite set of clauses. An interpretation \( J \) is a function, that maps each Boolean variable to a truth value \( \{\top, \bot\} \). A clause \( C \) is satisfied by an interpretation \( J \), if \( \exists l \in C : J(l) = \top \). An interpretation \( J \) satisfies a formula \( F \) in CNF, if all clauses in \( F \) are satisfied by \( J \). Two formulas \( F \) and \( F' \) are logically equivalent, denoted by \( F \equiv F' \), if they are satisfied by exactly the same interpretations. A formula \( F \) models a clause \( C \), if \( C \) is satisfied by all satisfying interpretations for \( F \), and is denoted by \( F \models C \). If a formula \( F \) models a clause \( C \), then \( F \equiv F \cup C \). The satisfiability problem is the problem to find a satisfying interpretation \( J \) for a given formula \( F \) in CNF or to prove that \( F \) cannot be satisfied by any interpretation. Answering this problem is \( \mathcal{NP} \)-complete [1].

2.2 Periodic Event Scheduling Problem

Several time scheduling problems have periodic properties, like the one presented in Section 4. Hence, a time event does not only happen once in time, but periodically often modulo a time bound. More in depth information can be found in [4].

Having two integers \( a, b \in \mathbb{Z} \), \([a, b] := \{x \in \mathbb{Z} | a \leq x \leq b\}\) denotes an interval from \( a \) to \( b \). We can extend this by a natural number \( t \in \mathbb{N} \), such that \([a, b]_t := \bigcup_{z \in \mathbb{Z}} [a + z \cdot t, b + z \cdot t] \subseteq \mathbb{Z} \) denotes an interval modulo \( t \).

Let \((V, \mathcal{E})\) be a directed graph, \( t \in \mathbb{N} \) and \( a : \mathcal{E} \rightarrow \mathcal{P}(\mathcal{P}(\mathbb{Z})) \) a mapping, that assigns to each edge \( e \in \mathcal{E} \) a set of intervals modulo \( t \). The tuple \( \mathcal{N} = (V, \mathcal{E}, a, t) \) is called periodic event network (PEN) with \( t \) being the period, \( V \) the set of (periodic) events, and \( a(e) \) the set of constraints for each edge \( e \in \mathcal{E} \).
Let $\mathcal{N} = (\{p,q,r,s\}, \mathcal{E}, a, 60)$ be a PEN with
\[
\mathcal{E} = \{(p,q), (p,s), (r,s)\}
\]
\[
a(p,q) = \{[1,3]_{60}\}
\]
\[
a(p,s) = \{[5,10]_{60}\}
\]
\[
a(r,s) = \{[1,2]_{60}\}.
\]

![Periodic event network with 4 events](image)

In the following, we regard always the PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$. In order to assign each event a point in time, we use a mapping $\Pi : \mathcal{V} \rightarrow \mathbb{Z}$ called schedule, which maps each event $n$ to an equivalence class $\Pi(n)$. The remaining work will always expect $\Pi, \Phi$ being schedules.

With $i, j \in \mathcal{V}$ being events, an edge $e = (i, j) \in \mathcal{E}$ holds under $\Pi$, denoted as $\Pi \models e$, if and only if $\forall \{l,u\} \in a(i, j) : \Pi(j) - \Pi(i) \in \{l,u\}$.

**Example 1.** Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN with $i, j \in \mathcal{V}$ and $e = (i, j) \in \mathcal{E}$ be an edge with $a(e) = \{[3,5]_{10}\}$.

- If $\Pi(i) = 3$ and $\Pi(j) = 8$, then $e$ holds, because $8 - 3 = 5 \in [3,5]_{10}$.
- If $\Pi(i) = 5$ and $\Pi(j) = 5$, then $e$ does not hold, because $5 - 5 = 0 \notin [3,5]_{10}$.
- If $\Pi(i) = 6$ and $\Pi(j) = 1$, then $e$ holds, because $1 - 6 = -5 \in [3,5]_{10}$.

In Figure 2 there are all feasible assignments with respect to $\Pi(i)$ and $\Pi(j)$ marked as blue circles, if $e$ holds. The reason, why each domain of $\Pi(i)$ and $\Pi(j)$ are finitely displayed ($\{0,\ldots,9\}$) is clarified with (2).

A constraint with respect to an edge $e = (i, j)$ defines a time consuming process, which describes, how much time within $[l,u]$ modulo $t$ is needed from an event $i$ to another event $j$, in order to hold. A schedule is said to be valid, denoted as $\Pi \models \mathcal{N}$, if the following equivalence holds
\[
\Pi \models \mathcal{N} : \iff \forall e \in \mathcal{E} : \Pi \models e
\]  

**Example 2.** Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, 60)$ be the PEN of Figure 1 and $\Pi$ a schedule of $\mathcal{N}$ with $\Pi = \{p \mapsto 144,\ q \mapsto 147,\ r \mapsto 148,\ s \mapsto 150\}$. $\Pi$ is valid with respect to $\mathcal{N}$, because the following conditions hold: $\Pi(q) - \Pi(p) = 147 - 144 = 3 \in [1,3]_{60}$, $\Pi(s) - \Pi(r) = 150 - 148 = 2 \in [1,2]_{60}$.

Investigating schedules for similarities, we can compare two schedules for equivalence. Two schedules $\Pi, \Phi$ are said to be equivalent, denoted as $\Pi \equiv \Phi$, if and only if $\forall n \in \mathcal{V} : \Pi(n) \mod t = \Phi(n) \mod t$. Knowing the mathematical behaviour of the $\mod$ operator, we can conclude, that any schedule $\Pi \in [\Phi]_{\equiv}$ within the equivalence class is valid, if $\Phi$ is valid, with respect to $\mathcal{N}$.

Thus, we can imply, that for each valid schedule $\Phi$, there exists an equivalent schedule $\Pi \in [\Phi]_{\equiv}$, such that
\[
\forall n \in \mathcal{V} : \Pi(n) \in [0,t - 1].
\]
With (2) the finite domain’s choice of Example 1 has been clarified. In the remaining work, it will always be tried to find the equivalence class’ member presented in (2). Of course, it is not known beforehand, which equivalence classes represent valid schedules for its members. Yet, it is sufficient to search for the described schedules for all equivalence classes of the quotient set, in order to find a valid schedule.

The periodic event scheduling problem (PESP) is the decision problem, whether a valid schedule with respect to a PEN does or does not exist.

Solving such a PESP instance can be done in several ways. One way is to solve it by a state-of-the-art PESP solver [2] as of Section 4 or with the method described in Section 3. It has been shown in [3], that PESP is \( \mathcal{NP} \)-complete.

### 3 Encoding PESP as SAT

This sections describes how a PESP can be reduced to a SAT problem. Similarly as converting constraint satisfaction problems into SAT, the finite domains of the events can be translated in very different ways. Due to the lack of space we present here only the order encoding \([5]\) in details. As shown in [7], the direct encoding can also be used for the translation. However, because domains in PESP are subsets of the natural numbers \( \mathbb{N} \), especially they are intervals, it is natural to apply the order relation \( \leq \).

Let \( x \in \mathcal{V} \) be an event with \( \text{dom}(x) = [l, u] \subset \mathbb{N} \). Then the function \( \text{enc} : \mathcal{V} \rightarrow \mathbb{CNF} \) is the order encoding function for an event. In the sequel, the propositional variable \( q_{x,i} \) represents that the value of the event \( x \) is less equal than \( i \).

**Definition 1 (Order Encoding Function for Variables of Finite Domains).**

\[
\text{enc} : x \mapsto (\neg q_{x,l-1} \land q_{x,u}) \bigwedge_{i \in [l,u]} (\neg q_{x,i-1} \land q_{x,i})
\]

It can be shown that for each interpretation \( J \) that models the encoding \( \text{enc}(x) \) of an event \( J \models \text{enc}(x) \), there exists exactly one value for the domain. Due to restricted space we refer to [2] for the proof. Extracting the value of the event \( x \) with the domain \( \text{dom}(x) = [l, u] \) from a given interpretation \( J \) is done by the function \( \xi_x(J) \), where \( \xi_x(J) = k, k \in [l, u] : J(q_{x,k-1}) = \bot \land J(q_{x,k}) = \top \).

In order to encode the potential of all nodes of \( \mathcal{N} \), we define

\[
\Omega_{\mathcal{N}} := \bigwedge_{n \in \mathcal{V}} \text{enc}(n), \tag{3}
\]

such that we can extract the schedule \( \Pi \) on a per-element basis from an interpretation \( J \), with \( J \models \Omega_{\mathcal{N}} \), by

\[
\forall n \in \mathcal{V} : \Pi(n) = \xi_n(J). \tag{4}
\]
3.1 Encoding the Constraints

To encode a constraint \( c = [l, u] \) of the edge \((i, j)\) we take a deeper look at all the feasible pairs \((\Pi(i), \Pi(j))\), which holds under \( c \). We call the sum of all these pairs the feasible region and the sum of every other pair the infeasible region. Figure 2 displays the feasible region for a constraint \([3, 5]_{10}\) with a black shape.

Since the value of an event is order encoded, we can define a rectangle \((x_1, y_1) \times (x_2, y_2)\) in the infeasible region, written as \([x_1, x_2] \times [y_1, y_2]\), with a single clause:

\[
\text{enc}
\text{rec}([x_1, x_2] \times [y_1, y_2]) = \neg q_{x,x_1} q_{x,x_2}, \neg q_{y,y_2} q_{y,y_1-1}
\]

To define a rectangle for the feasible region we would need more than a single clause, hence we decided to encode a constraint by defining the infeasible region with rectangles to keep the number of clauses low. Empirical results indicates that a lower number of clauses typically results in a lower run time behavior of a SAT solver.

First of all, we need some helper functions. These helper functions will be concluded in a function, which calculates as few rectangles as possible to cover the whole not feasible region of a constraint. This results in the lowest possible number of clauses, which should help a SAT solver to find a solution faster having less redundancy.

Example 3. Let \( N = (V, E, a, t) \) be a PEN with \( i, j \in V \) and \( e = (i, j) \in E \) be an edge with \( a(e) = \{3, 5\}_{10} \). A not feasible region is for example \( r = ([4, 7] \times [3, 6]) \) with \((\Pi(i), \Pi(j)) \notin r\). The rectangle \( r \) that is excluded is visualized in Figure 2. \( \{ (m, n) \mid m \in [4, 7], n \in [3, 6] \} = r \) is the set of all pairs that are not feasible. Then, we know, that for each pair \((\Pi(i), \Pi(j)) \notin r\), such that \( e \)
holds for $\Pi(i), \Pi(j)$. With $q_{i,n}, n \in [-1,9]$ and $q_{j,m}, m \in [-1,9]$ being the propositional variables of Definition 1, it follows
\[\#(\Pi(i), \Pi(j)) : \Pi(i) \leq 7, \Pi(i) \geq 4, \Pi(j) \leq 6, \Pi(j) \geq 3\]
\[\Leftrightarrow \neg((\Pi(i) \leq 7) \land (\Pi(i) \geq 4) \land (\Pi(j) \leq 6) \land (\Pi(j) \geq 3))\]
\[\Leftrightarrow \neg(q_{\Pi(i),7} \land q_{\Pi(i),3} \land q_{\Pi(j),6} \land \neg q_{\Pi(j),2})\]
\[= [-q_{\Pi(i),7}, q_{\Pi(i),3}, q_{\Pi(j),6}, q_{\Pi(j),2}] := F\]
Since $F$ is a clause, we can directly append this to the resulting formula.

Reconsidering Figure 2 displays the two feasible regions. Each of which has a lower and an upper bound. In order to describe the not feasible region in between, we need the next lower bound, which represents the upper feasible region and the previous lower bound, which represents the lower feasible region with respect to $\Pi(j)$. With respect to Figure 2 and the displayed not feasible region, we would have an upper bound $u = -5$ and a lower bound $l = 3$. It can be directly concluded, that $u < l$.

With $\lfloor \cdot \rfloor$ being the round down function, $\lceil \cdot \rceil$ being the round up function and the two integers $u, l \in \mathbb{Z}$ with $u < l$, we can define
\[\delta : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, \delta(l, u) = l - u - 1\]
\[\delta y : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, \delta y(l, u) = \left\lfloor \frac{\delta(l, u)}{2} \right\rfloor\]
\[\delta x : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, \delta x(l, u) = \left\lceil \frac{\delta(l, u)}{2} \right\rceil - 1.\]

With these definitions, we are able to determine a rectangle between $u$ and $l$, such that it has maximum area having minimum perimeter.

Basically each rectangle has a width of $\delta x(l, u)$ and a height of $\delta y(l, u)$. In order to cover each not feasible pair in the area between $u$ and $l$, we need approximately $t$ rectangles. The possible worse runtime behavior of other shapes will not be verified in this work.

Let $P_I := \{ (\Pi(i), \Pi(j)) \mid \Pi(j) - \Pi(i) \notin I \}$ be the set of pairs, which are part of the not feasible region of constraint $I$ with respect to edge $(i, j)$. Then $S_I := [0, t - 1] \times [0, t - 1] \setminus P_I$ is the set of feasible pairs of $I$.

The function $\zeta : \mathcal{P}(\mathbb{Z}) \rightarrow \mathcal{P}(\mathcal{P}(\mathbb{Z}) \times \mathcal{P}(\mathbb{Z}))$,
$\zeta([l, u]_I) = \{ A \times B \in \mathcal{P}(\mathbb{Z})^2 \mid |A| = \delta x(l, u), |B| = \delta y(l, u), (A \times B) \cap S_I \cap [l, u], = \emptyset \}$,
with $A, B \in \mathcal{P}(\mathbb{Z})$ being intervals, maps to the set of all not feasible rectangles of constraint $I$ with respect to edge $e$. The sufficiency, that all not feasible pairs of a constraint are excluded, is given by the following lemma.

Lemma 1. Let $I = [l, u]_I \in a(e)$ be a constraint of edge $e$. Then, it holds
\[(i) \quad P_I \subseteq \bigcup_{A \in \zeta(l)} A\]
\[(ii) \quad S_I \cap \bigcup_{A \in \zeta(l)} A = \emptyset\]
Proof. In technical report.

Like in Example 3, the function $\text{enc}_{\text{rec}} : \mathcal{P}(\mathbb{Z}) \times \mathcal{P}(\mathbb{Z}) \rightarrow \mathcal{L}(\mathcal{R})$ maps to order encoding for the to be excluded rectangle, such that $\text{enc}_{\text{rec}}([x_1, x_2] \times [y_1, y_2]) = \{-q_{x,x_2}, q_{x,x_1+1}, -q_{y,y_2}, q_{y,y_1+1}\}$.

Lemma 2. Let $A = ([x_1, x_2] \times [y_1, y_2]) \in \mathcal{P}(\mathbb{Z}) \times \mathcal{P}(\mathbb{Z})$ be a rectangle with $(x, y) \in A$ and $q_{l,k} \in \mathcal{R}$ ($l \in \{x, y\}, k \in \text{dom}(l)$) being the propositional variables of Definition 1. Then

$$J \models \text{enc}_{\text{rec}}(A) \iff (\xi_x(J), \xi_y(J)) \notin A$$

with $J$ being an interpretation and $\xi_n(J)$ being the extracted potential of (??).

Proof. In technical report.

Let $\Psi_N$ being the encoding of all edges $e \in \mathcal{E}$, such that

$$\Psi_N := \bigwedge_{e \in \mathcal{E}} \bigwedge_{I \in a(e)} \bigwedge_{A \in \xi(I)} \text{enc}_{\text{rec}}(A),$$

we can encode a PESP instance with

$$\text{enc}_{\text{pesp}}((V, \mathcal{E}, a, t)) = (\Omega_N \land \Psi_N)$$

Theorem 1 (Soundness and Completeness).
Let $\mathcal{N} = (V, \mathcal{E}, a, t)$ be a PEN and $F := \text{enc}_{\text{pesp}}(\mathcal{N}) \in \mathcal{L}(\mathcal{R})$ be the order encoded propositional formula of $\mathcal{N}$. Then

$$\exists J : J \models F \iff \exists \Pi : \Pi \models \mathcal{N}$$

with $J$ being an interpretation and $\Pi$ a schedule of $\mathcal{N}$. 
Proof.

\[ \exists J : J \models F \]

(6) \[ \exists J : J \models (\Omega_N \land \Psi_N) \]

\[ \iff \exists J : J \models \Omega_N \land J \models \Psi_N \]

(3) \[ \exists J : J \models \bigwedge_{n \in \mathcal{V}} \text{enc}(\Pi(n)) \land J \models \Psi_N \]

\text{Lem} \, \text{(4)} \iff \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_{\Pi(n)}(J) \land J \models \Psi_N

(5) \[ \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_{\Pi(n)}(J) \]

\[ \land J \models \bigwedge_{e \in \mathcal{E}} \bigwedge_{I \in a(e)} \text{enc}_{\text{con}}(e, I) \]

\[ \iff \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_{\Pi(n)}(J) \]

\[ \land \forall e \in \mathcal{E} : \forall I \in a(e) : J \models \text{enc}_{\text{con}}(e, I) \]

\text{Lem} \, 2, \text{Lem} \, 1 \iff \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_{\Pi(n)}(J) \land \forall e \in \mathcal{E} : e \text{ holds under } \Pi

\text{Def} \, 1 \iff \exists J : \forall n \in \mathcal{V} : \Pi(n) := \xi_{\Pi(n)}(J) \land \Pi \models N

\text{Lem} \, \text{(7)} \iff \exists \Pi : \Pi \models N

\square

4 Results

5 Conclusion

In this work it has been shown, that we can successfully reduce a PESP instance to a SAT instance and solve it with a state-of-the-art SAT solver in a very short time frame compared to a state-of-the-art PESP solver.

It has been shown that SAT-based approach outperforms by far the traditional PESP solver. Thus, this new method can be applied to a whole new set of larger instances, which could not have been solved before.

Several preprocessing methods could be applied to the periodic event network, in order to encode more information into the SAT instance, namely, propagating gained information of constraints to neighboured constraints and cutting search space of the potentials, which are not part of the solution space. This may help the SAT solver to find faster a solution. In addition, the instance can be shortened by removing redundant constraints.

Furthermore, it should be tried to solve even larger and harder instances. The largest possible instance for the railway modeling topic would be a union of all
Table 1. PESP instances and corresponding encodings

<table>
<thead>
<tr>
<th>instance</th>
<th>PESP $\mathcal{N} = (V, E, a, t)$</th>
<th>direct encoding $F$</th>
<th>order encoding $G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>$18$</td>
<td>$730$</td>
<td>$2,160$</td>
</tr>
<tr>
<td>$swg_2$</td>
<td>$60$</td>
<td>$1,145$</td>
<td>$7,200$</td>
</tr>
<tr>
<td>$fernsym$</td>
<td>$128$</td>
<td>$3,117$</td>
<td>$15,360$</td>
</tr>
<tr>
<td>$swg_4$</td>
<td>$170$</td>
<td>$7,107$</td>
<td>$20,400$</td>
</tr>
<tr>
<td>$swg_3$</td>
<td>$180$</td>
<td>$2,998$</td>
<td>$21,600$</td>
</tr>
<tr>
<td>$mb11mg$</td>
<td>$200$</td>
<td>$8,298$</td>
<td>$23,280$</td>
</tr>
<tr>
<td>$b$</td>
<td>$219$</td>
<td>$5,571$</td>
<td>$22,320$</td>
</tr>
<tr>
<td>$swg_1$</td>
<td>$221$</td>
<td>$7,443$</td>
<td>$26,520$</td>
</tr>
<tr>
<td>$mb13mg$</td>
<td>$231$</td>
<td>$9,805$</td>
<td>$27,000$</td>
</tr>
<tr>
<td>$save$</td>
<td>$277$</td>
<td>$1,535$</td>
<td>$21,000$</td>
</tr>
<tr>
<td>$seg_2$</td>
<td>$611$</td>
<td>$9,863$</td>
<td>$73,320$</td>
</tr>
<tr>
<td>$we$</td>
<td>$846$</td>
<td>$14,690$</td>
<td>$78,480$</td>
</tr>
<tr>
<td>$seg_1$</td>
<td>$1,483$</td>
<td>$10,351$</td>
<td>$177,960$</td>
</tr>
</tbody>
</table>

Table 2. PESP instances and corresponding times to solve

<table>
<thead>
<tr>
<th>instance</th>
<th>pepsolve /s</th>
<th>direct + riss /s</th>
<th>ordered + riss /s</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>$swg_3$</td>
<td>$66$</td>
<td>$50$</td>
<td>$2$</td>
<td>$33$</td>
</tr>
<tr>
<td>$swg_2$</td>
<td>$512$</td>
<td>$37$</td>
<td>$2$</td>
<td>$256$</td>
</tr>
<tr>
<td>$swg_4$</td>
<td>$912$</td>
<td>$752$</td>
<td>$8$</td>
<td>$114$</td>
</tr>
<tr>
<td>$fernsym$</td>
<td>$2,035$</td>
<td>$294$</td>
<td>$7$</td>
<td>$290$</td>
</tr>
<tr>
<td>$swg_1$</td>
<td>timeout</td>
<td>$18$</td>
<td>$7$</td>
<td>$&gt;12,342$</td>
</tr>
<tr>
<td>$seg_1$</td>
<td>timeout</td>
<td>$16$</td>
<td>$10$</td>
<td>$&gt;8,640$</td>
</tr>
<tr>
<td>$seg_2$</td>
<td>timeout</td>
<td>timeout</td>
<td>$11$</td>
<td>$&gt;7,854$</td>
</tr>
<tr>
<td>$mb11mg$</td>
<td>timeout</td>
<td>-</td>
<td>$110s$</td>
<td>$&gt;785$</td>
</tr>
<tr>
<td>$b$</td>
<td>timeout</td>
<td>-</td>
<td>$145s$</td>
<td>$&gt;595$</td>
</tr>
<tr>
<td>$mb13mg$</td>
<td>timeout</td>
<td>-</td>
<td>$986s$</td>
<td>$&gt;87$</td>
</tr>
<tr>
<td>$we$</td>
<td>timeout</td>
<td>-</td>
<td>timeout</td>
<td>-</td>
</tr>
<tr>
<td>$k$</td>
<td>$47,112$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$save$</td>
<td>timeout</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

subject to a railway network of whole Germany. If it could be achieved to solve such an instance in an adequate time frame, a huge goal would be achieved with respect to solving a single instance for feasibility.

Another interesting part, the optimization of timetables, which has been already discussed a lot for example in [4], could be supported by a fast feasibility checking method, like presented in this work. In the process, several constraints of the periodic network with respect to its bounds and an extracted objective functional will be changed. This could be achieved by an adapted branch and bound algorithm.

Finally, it can be concluded, that the currently best PESP solver is now SAT-based.
References