Optimisation of Periodic Train Timetables
Extended abstract

Gonçalo Polónia de Matos

Abstract In this research work we address the optimisation of real-world, large-scale periodic train timetables with respect to three distinct objectives: minimising the total travel time, maximising the timetable robustness (i.e. the timetable capacity to absorb delays and prevent their propagation when subjected to disturbances), and minimising the amount of relaxation effectively used while obtaining approximate solutions for infeasible problems (whose constraints are previously relaxed).

We propose two approaches to this problem, both supported by a SAT solver. One is an exact method based on a binary search and an heuristic which, combined, produce optimal solutions. The other is an approximate method which uses reinforcement learning in conjunction with SAT to obtain optimised solutions for bigger problems.

The results that we present highlight the strengths and weaknesses of each method and validate their applicability to the three aforementioned objectives. Furthermore, a performance comparison with other state-of-the-art approaches is given. Our results suggest that the joint potential of machine learning and SAT techniques in this field is still in its beginnings but already providing up-and-coming outcomes, actually positioning the results of our method amongst the state-of-the-art on some publicly available benchmarking problems.

Keywords Periodic railway timetabling · Optimisation · Periodic Event Scheduling Problem · SAT · Reinforcement Learning

Mathematics Subject Classification (2000) 90B06 · 68T20 · 68T05

1 Introduction

Timetabling is a major step in the overall process of planning the operations in a public transport network, and is closely related to other planning activities such as line planning, and vehicle and crew scheduling.

In this work, we focus on periodic timetabling for public transport. In a periodic timetable, each event (e.g. a train departure from a certain station) scheduled for some time instant actually occurs recurrently at that same instant, at each period (e.g. every hour).

Concerning periodic timetables, one can be interested in finding a feasible timetable or an optimised one, according to some criteria. Henceforth, we will refer to these problems as Periodic Timetable Satisfaction Problem (PTSP) and Periodic Timetable Optimisation Problem (PTOP), respectively.

Over the last years, several approaches were studied for modelling and solving either the PTSP or the PTOP, namely constraint propagation, Mixed Integer Linear Programming (MILP), heuristically-guided search, genetic algorithms and the modulo network simplex [Nachtigall and Opitz (2008a)]. A well-known approach for modelling the PTSP, called the Periodic Event Scheduling Problem (PESP), was introduced by Serafini and Ukovich (1989).

Since PESP models were proposed, several approaches were used to solve the problem with increasing success. Recent works [Großmann et al (2012a), Großmann (2011), Kümmling et al (2015)] suggest that currently the best known approach for solving the PESP and the PTSP is to model them as a Boolean Satisfiability Problem (SAT)\(^1\). Essentially, the PESP problem is encoded

\(^1\) The Boolean Satisfiability Problem (SAT) is to decide whether a propositional formula is satisfiable or not. It is a well known and long time studied problem [Cook (1971)]
as a propositional formula and solved using a state-of-the-art SAT solver.

Besides solving the PTSP, which is challenging per se, solving the PTOP with an objective function such as minimising the travel time is also a major area of interest. Several works [Peeters (2003), Ingolotti et al (2006), Goerigk and Liebchen (2017)] have been done in this area, some of which also used SAT approaches [Großmann et al (2012b), Großmann et al (2015), Gattermann and Nachtigall (2016)].

This paper presents new developments on a research [Matos et al (2017)] performed in the context of a Master’s thesis [Matos (2017)] at Instituto Superior Técnico\(^2\) and SISCOG\(^3\).

In particular, we are interested in optimising the total passengers’ travel time and the timetable robustness\(^4\) (two linear objective functions defined by Peeters (2003)) using incremental calls to a SAT solver, on real-world, large-sized timetabling problems. We are also interested in addressing infeasible instances, minimising the amount of relaxation needed to solve conflicts.

In this paper, we propose a machine learning based method to guide the search, something which (to our knowledge) was never done before in this domain. We evaluate the performance of our solution by comparing our achieved results (in terms of objective value) on a publicly available set of problems with the ones obtained by other authors of state-of-the-art approaches to the PTOP. Furthermore, we also present our previously developed approach [Matos (2017)], based on incremental SAT calls, a binary search procedure and an heuristic to compute better upper bounds, and compare its performance with our new method.

On Section 2, we define the problem that we will be solving, after introducing the necessary background. Sections 3 and 4 describe, respectively, the proposed solution for this problem and some promising results which show that our approach performs better than the currently state-of-the-art approaches on some instances. Finally, we present conclusions (Section 5).

---

\(2\) The University of Lisbon engineering faculty.

\(3\) SISCOG - Sistemas Cognitivos, SA (http://www.siscog.pt), a Portuguese software company specialised in solutions for creating and updating operational plans of companies that provide regular transportation services, such as passenger railway operators and metro systems.

\(4\) A robust timetable is a timetable that does not tend to become disrupted when subjected to disturbances.

---

2 Definitions

2.1 The Periodic Timetable Satisfaction Problem (PTSP)

A Periodic Timetable Satisfaction Problem (PTSP) can be modelled as a Periodic Event Scheduling Problem (PESP) [Serafini and Ukovich (1989)] whose events correspond to periodic train/bus departures (arrivals) from (to) specific stations in the context of specific train/bus trips (see Figure 1). The goal is to find a schedule for such a set of periodically recurring events, subjected to a set of constraints (for instance, safety headways, trip and dwell times) under periodic time windows, or to conclude that no such assignment exists.

Any constraint between a pair of events \((i,j)\), given the corresponding scheduled times \(v_i\) and \(v_j\), has the form:

\[
v_j - v_i + Tp_{ij} \in [l_{ij}, u_{ij}]
\]

(1)

where \(p_{ij}\) is an auxiliary integer decision variable, \(T\) is the period of the timetable, and \(l_{ij}\) and \(u_{ij}\) are respectively the lower and upper bound, i.e., the minimum and maximum admissible values for that constraint.

2.2 The Periodic Timetable Optimisation Problem (PTOP)

The Periodic Timetable Optimisation Problem (PTOP) is to find an optimal solution to a PTSP, that is, to select from the set of all feasible solutions to a PTSP one that minimises (or maximises) a provided objective function.

One of the objective functions presented by Peeters (2003), which will be the main reference for this paper, is minimising the total travel time. This objective function depends on process times, which can be defined as:

\[
x_{ij} \equiv v_j - v_i + Tp_{ij}
\]

(2)
Connection, dwell and trip times influence the total passenger travel time. For instance, each minute of dwell above the minimum dwell time (denoted by \( l_{ij} \)) adds to the minimum possible travel time. So, these excess process times are given by \( (x_{ij} - l_{ij}) \).

If we define a weight \( w_{ij} \) for each constraint, then we want to minimise:

\[
\sum_{(i,j) \in A} w_{ij}(x_{ij} - l_{ij})
\]

(3)

### 3 Solution architecture

#### 3.1 Overview

In this work, we propose two new approaches to address the PTOP. One is an exact method — which guarantees the optimal solution — based on a binary search procedure and supported by an heuristic to compute a good initial upper bound. The other is a method which combines SAT with machine learning to get optimised solutions iteratively for bigger problems. On both approaches we used a SAT solver and modelled the PTOP in SAT using the order encoding approach suggested by Großmann (2011).

The overall architecture is summarised in Figure 2. The block “Create PEN graph” consists in transforming the provided PTOP into a Periodic Event Network (PEN) graph using the tools already available in ONTIME\(^5\). The block “Compaction” represents a set of graph operations that were already implemented in ONTIME in order to reduce the PEN as much as possible before actually starting solving the problem (for instance, the contraction of edges with a fixed length, or the merge of multiple parallel edges (constraints) between the same pair of events). The “Order encode PTSP” procedure consists in simply encoding the satisfaction problem with the order encoding. The block “Encode objective function” represents the encoding of the optimisation part, which will be explained in Section 3.2. The reason we separate the encoding of the satisfaction and optimisation parts of the problem is because the choice of the objective function may be dependent on the existence of a satisfiable solution. The block “Incremental SAT optimisation” is a black box representing any of the two proposed methods in this thesis to find optimised solutions to the PTOP — either the binary search procedure (Section 3.3) or the machine learning (ML) approach (Section 3.4 and Section 3.5). Finally, “Extract timetable” consists in getting a PTOP solution — i.e., an assignment of schedule times to events in the PEN graph — from the optimised SAT solution.

To improve a solution with respect to the objective function (Equation 3) we have to reduce some process times. However, in order to be able to reduce a process time, one usually needs to increase another process time to satisfy the constraints and get a SAT answer, since implicit relations exist between PTOP constraints. The great difficulty of this process comes from the fact that we usually do not know explicitly those relationships.

Therefore, in order to find an optimised solution, our two proposed procedures need to perform a good balance between process times.

#### 3.2 Encoding the objective functions into SAT

Our first step towards solving the PTOP, either with the binary search or the machine learning approach, is to encode the problem into SAT. First, we encode the underlying PTSP problem using the order encoding, as described in Großmann (2011) for PESPs. We describe next how we encode the objective function, which is (in the case of this short paper) to minimise the total travel time.

With that purpose, we add auxiliary variables \( \tau^k_{ij} \in \{0, 1\} \), for all constraints \((i, j) \in A \) and \( k \in \{0, ..., u_{ij} - l_{ij}\} \), representing the difference between the tension \( x_{ij} \) and the lower bound \( l_{ij} \) for the constraint, such that the variable \( \tau^k_{ij} \) is true if the proposition \( v_j - v_i - l_{ij} + T \cdot p_{ij} \geq k \) holds, being \( p_{ij} \in \mathbb{Z} \) auxiliary decision variables.

Then, we impose order encoding constraints on those \( \tau \) variables:

\[
(i, j) \mapsto \big( \tau^0_{ij} \land \tau^u_{ij} + 1 \land \bigwedge_{k \in \{1, ..., u_{ij} - 1\}} (\neg \tau^k_{ij} \lor \tau^{k-1}_{ij}) \big)
\]

(4)

and encode the following implications:

\[
(v_j - v_i - l_{ij} + T \cdot p_{ij} \geq k) \implies \tau^k_{ij}
\]

(5)

which can be written with a disjunction

\[
(v_j - v_i + T \cdot p_{ij} \in [l_{ij}, k + l_{ij} - 1]) \lor \tau^k_{ij}
\]

(6)

\(^5\) One of SISCOG Suite’s products. For more information, access the ONTIME product page on SISCOG’s web site (http://www.siscog.pt).
Finally, we want to minimise the total travel time, i.e.:

\[ \sum_{(i,j) \in A} (x_{ij} - l_{ij}) = \sum_{(i,j) \in A} \left( \sum_{k=1}^{u_{ij} - l_{ij}} \tau_{ij}^k \right) \quad (7) \]

This last expression is called a pseudo-boolean expression, and it can be converted into clausal form. Therefore, we can address this optimisation problem with a SAT solver.

### 3.3 Optimising with SAT and a binary search approach

The first approach that we propose to get the optimal solution for a PTOP is to perform a binary search on the value of the objective function, supported by a lower bound (LB) and an upper bound (UB) that we compute beforehand.

As we saw before, our goal is to minimise an objective function in the form of a pseudo-boolean expression. Henceforth, we model the function as a constraint instead, with the general form:

\[ \sum_i w_i \cdot b_i \leq C \quad (8) \]

being \( C \) an integer constant, \( b_i \) the boolean variables and \( w_i \) their respective coefficients (weights). Then, we apply a binary search procedure to replace \( C \) by actual integer values and perform iterative SAT calls in search for the optimal solution.

In order to compute a better UB to begin with the binary search, and thus improve the solving time, we developed an heuristic which uses information from UN-SAT cores\(^6\) (see Algorithm 1).

Further details on this approach are given in Matos et al (2017).

### 3.4 Modeling the problem with agents and reinforcement learning

Our first algorithmic solution to the problem suffers from a big flaw — it has no idea about the relations between the PTOP constraints implicitly present in the problem and thus is too greedy. Therefore, after the binary search procedure, we developed an approach based on a multi-agent, reinforcement learning system to solve the PTOP trying to “guess” the relationships between constraints based on data from hundreds of experiments. In the end, this method aims to get a good PTOP solution (although the optimal one cannot be guaranteed), performing a good balance — inferred by experience — between decreased and increased process times. The results obtained with this new approach were successful enough to position it among the state-of-the-art and to motivate our submission of a paper to the 14th International Conference on Advanced Systems in Public Transport (CASPT), which was indeed accepted and will be soon available [Matos et al (2018)].

In this approach, we associate to each process time \( x_{ij} \) an agent whose possible actions are adding constraints \( x_{ij} \leq u \) (where \( u \) varies from \( l_{ij} \) to \( u_{ij} \)) to

---

\( ^6 \) An UNSAT core is a sub-set of clauses from the SAT problem which is unsatisfiable. UNSAT cores can give insightful information about the problem being solved, and thus techniques have been developed to extract and exploit them (for instance, Fu and Malik (2006)).
the original formulation, which, after being encoded in SAT, result in a set of binary formulas called assumptions. Henceforth, each agent will determine a set of SAT assumptions to be added on top of the original formulation, which will further constrain the problem and guide the search.

The architecture of this approach is based on a decide–act–observe reward cycle such that all agents are given the opportunity to decide on their action, then all the actions are performed simultaneously and influence the environment — i.e., a solution for the PTSP is produced with some cost —, and finally all the agents receive feedback on how well they performed so that they can improve and learn from their mistakes and/or from their good decisions.

Agents receive feedback by the means of a reward function that we must define, which should provide an individual value for each agent based on how good or bad its decision was. The reward function developed in this work is a function with two terms:

- A global term — which rewards the agent accordingly to the overall solution cost, i.e., the lower the solution cost is relative to the initial solution value the greater is the reward the agent receives;

  \[
  \text{global\_term} = \frac{UB_{\text{initial}} - \text{solution\_cost}}{UB_{\text{initial}}} \tag{9}
  \]

- An individual term — which encourages the agent to choose actions which correspond to lower tension values since, in general, the lower the edge tensions are the lower the solution cost is;

  \[
  \text{individual\_term} = \frac{\text{agent\_NrActions} - \text{agent\_action\_chosen}}{\text{agent\_NrActions}} \tag{10}
  \]

3.5 The learning algorithm

The learning algorithm we implemented is based on the Upper Confidence Bound (UCB1) [Auer et al (2002)] action selection method, with some ideas from the Q-learning algorithm [Watkins and Dayan (1992)] along with other small improvements that take advantage from knowledge from the structure of a PTOP problem.

In each cycle, every agent is given the opportunity to choose an action. Then, the actions of all agents are combined, i.e. each agent sets an upper bound to the corresponding process time, by means of the assumptions feature of the SAT solver. Next, the SAT solver is called to test whether a solution to the PTOP with such upper bounds for the process times exists and, in the affirmative case, to obtain a model (a feasible assignment of times to the events).

According to the answer returned by the SAT solver, a reward is computed and provided for each agent. If the answer was SAT, then a reward is computed according to the aforementioned reward function. If, otherwise, the answer was UNSAT, then an UNSAT core is extracted and the agents corresponding to the constraints involved in the core receive 0 as a reward (a penalty), whilst the remaining agents do not receive any reward at all and hence do not learn anything (as if their action never occurred) because we cannot know for sure whether their decisions were good or bad. The introduction of knowledge from the UNSAT cores is one of the particularities of our learning algorithm.

After an initialisation period in which each agent tries all its actions once, the action for the time-step \( t \) is chosen according to the formula:

\[
\text{action}_t = \arg\max_{a \in \text{agent\_Actions}} QValues[a] + \sqrt{\frac{\log(t)}{\text{NumTrials}[a]}} \tag{11}
\]

where \( \text{NumTrials} \) is the total number of times the action was performed in the past, and \( \alpha \) is the diversification parameter of the algorithm. In our experiments, we found that the value 0.001 worked well for this parameter.

After executing an action \( a \) and incrementing the corresponding \( \text{NumTrials}(a) \) counter, an agent updates the Q-value for the action \( a \), given a reward, using the formula:

\[
QValues[a]' = QValues[a] \times (\text{NumTrials}[a] - 1) + \text{reward} \div \text{NumTrials}[a] \tag{12}
\]

which essentially keeps an average of the rewards obtained in the past when that action was chosen.

After thousands of iterations, the agents’ behaviour converges to an optimised solution (see Figure 3).

Fig. 3 Behaviour of the machine learning approach when solving the R4L1 problem from PESPlib. Solution values (y axis) across algorithm iterations (x axis). Lower values are better.

4 Results and discussion

All the tests in the following sections were run on a machine with an Intel Core i7-6700K CPU @ 4.00GHz.
and 64GB of RAM. We used the SAT solver Crypto-minisat 5.0.1 [Soos (2016)] with one thread, the Maximum Satisfiability Problem (MaxSAT) solver Open-WBO 1.3.1 [Martins et al (2014)], the MILP solver IBM ILOG CPLEX 12.7, and a Constraint Programming (CP) solver developed from scratch at SISCOG and embedded in ONTIME.

### 4.1 Comparison between SAT and other approaches

For this experiment, we applied our SAT-based binary search approach to real world data from a major European passenger railway operator and a major US metro system, and we used the travel time as objective function. We used the binary search approach with the core guided heuristic for this experiment, since we wanted an exact method to make a fair comparison with the other approaches (which are also exact methods).

Table 1 shows the execution times for finding an optimal solution and the value of the best solution found with each approach. To evaluate the performance of our proposed method (which uses a SAT approach), we compared it with the execution time needed for an equivalent MaxSAT approach\(^8\), with the execution time needed for the MILP model\(^7\) used in ONTIME, and with a CP approach\(^9\) also linked to ONTIME. In Table 1, the execution time entries with a + represent runs which were interrupted when the time limit of two hours that we imposed was achieved (in those cases, the values shown on Table 1 are the best found up to that time). Table 1 also shows the gap to optimality computed by each method, taking into account the best LB it could find. In SAT and in ONTIME’s CP there is no obvious way of computing LBs, since these approaches rely on a satisfaction (not optimisation) solver that iteratively finds solutions with lower values. Therefore, the best we can do in those cases is to consider the naive and lowest solution value that is possible for any timetabling problem, which is 0.

The results show that our approach outperforms the MILP approach, running on the commercial CPLEX solver, in most of the problems. For several of them, we could even stop the execution of CPLEX after a couple of hours without having found any solution at all, while our SAT based approach found the optimal solution in just a few seconds. The bad results of the MILP solver may be due to poor LBs.

Furthermore, the results also show that our approach outperforms an equivalent MaxSAT approach, since the execution times were lower for all problems with no exception. We observed that the MaxSAT solver had great difficulty in finding good LBs for this problem (hence the big gaps), frequently becoming stuck for long periods of time, perhaps testing the feasibility of some sub-optimal solutions and being unable to find out that they were unsatisfiable (UNSAT). Therefore, we believe in the hypothesis that our specialised solver based on incremental SAT calls coupled with a meta-heuristic search procedure can outperform a more general MaxSAT solver.

However, our method also has its limitations. The results for the problems \(R1\) and \(R2\) were obtained with the core guided heuristic only, i.e. not followed by the binary search, since we could not make the encoding of the pseudo-boolean constraints needed to perform the binary search fit in the available 64GB of memory. This fact also explains why we have such a bad gap on those two problems — since we could not use the binary search, the best LB we have is the naive 0. Nonetheless, we were still happy to find out that our core guided heuristic found the optimal solution\(^10\) for problem \(R1\) on its own, without the need for the binary search procedure. Moreover, on the metro problems the binary search was not actually executed either, not due to memory limitations, but because the heuristic found by itself the best possible solution, which has value 0. In other experiments in the context of the research, though, we had the opportunity to see the binary search and each one of our proposed methods in action. For this first experiment we were not particularly concerned about the specificities of the solving method; comparing the core technology (i.e. SAT) to other approaches was all we wanted.

### 4.2 Benchmarking the machine learning approach

For benchmarking purposes, we compared the performance of our machine learning approach on a set of PTOP problems freely available as part of the PESPlib\(^11\), created with the LinTim toolbox [Goerigk and Schöbel (2013)], with the results also published in that library’s web page. The PESPlib consists on a set of railway and bus timetabling problems formulated as PESPs. The best known solutions in terms of travel time are regularly published in the corresponding web

---

\(^7\) Which we also implemented.

\(^8\) Based on Peeters (2003).

\(^9\) Based on de Waal (2005).

\(^10\) We know this because the MILP solver found the same solution and proved its optimality. Without the binary search procedure, our core guided heuristic alone cannot prove the optimality of the solution.

\(^11\) [http://num.math.uni-goettingen.de/~m.goerigk/pesplib/](http://num.math.uni-goettingen.de/~m.goerigk/pesplib/)
Table 1 Comparison between SAT, MaxSAT, MILP and CP approaches on real world data. Lower values are better.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Gap %</th>
<th>TIME</th>
<th>Value</th>
<th>Gap %</th>
<th>TIME</th>
<th>Value</th>
<th>Gap %</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>39 601</td>
<td>100.00</td>
<td>00:00:11</td>
<td>39 629</td>
<td>96.22</td>
<td>02:00:00</td>
<td>39 601</td>
<td>100.00</td>
<td>00:00:27</td>
</tr>
<tr>
<td>R2</td>
<td>38 059</td>
<td>100.00</td>
<td>00:00:30</td>
<td>45 558</td>
<td>99.24</td>
<td>02:00:00</td>
<td>38 993</td>
<td>100.00</td>
<td>02:00:00</td>
</tr>
<tr>
<td>M1-24</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:14</td>
<td>0.00</td>
<td>0.00</td>
<td>02:00:00</td>
<td>0.00</td>
<td>0.00</td>
<td>02:00:00</td>
</tr>
<tr>
<td>M2-48</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:05</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:15</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:47</td>
</tr>
<tr>
<td>M3-80</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:14</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:32</td>
<td>- -</td>
<td>- -</td>
<td>00:00: +</td>
</tr>
<tr>
<td>M4-96</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:14</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:44</td>
<td>- -</td>
<td>- -</td>
<td>00:00: +</td>
</tr>
<tr>
<td>M5-112</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:24</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:55</td>
<td>- -</td>
<td>- -</td>
<td>00:00: +</td>
</tr>
<tr>
<td>M6-120</td>
<td>0.00</td>
<td>0.00</td>
<td>00:00:22</td>
<td>0.00</td>
<td>0.00</td>
<td>00:01:01</td>
<td>- -</td>
<td>- -</td>
<td>00:00: +</td>
</tr>
</tbody>
</table>

Table 2 Benchmarking with PESPlib’s problems — our results on all problems. In bold our own contributions to the web page. Lower values are better.

<table>
<thead>
<tr>
<th>Name</th>
<th>Events</th>
<th>Constraints</th>
<th>Previously published value (on 26/09/2017)</th>
<th>Our value (on 3/10/2017)</th>
<th>Best published value (on 05/02/2018)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1L3</td>
<td>3 064</td>
<td>4 396</td>
<td>31 858 161</td>
<td>31 858 161</td>
<td>31 858 161</td>
</tr>
<tr>
<td>R1L2</td>
<td>3 069</td>
<td>6 544</td>
<td>38 248 406</td>
<td>41 708 862</td>
<td>41 708 862</td>
</tr>
<tr>
<td>R1L1</td>
<td>4 184</td>
<td>7 032</td>
<td>38 612 098</td>
<td>38 160 248</td>
<td>38 160 248</td>
</tr>
<tr>
<td>R1L4</td>
<td>4 760</td>
<td>8 529</td>
<td>35 955 823</td>
<td>35 955 823</td>
<td>36 055 913</td>
</tr>
<tr>
<td>R1L4</td>
<td>4 156</td>
<td>7 362</td>
<td>53 708 802</td>
<td>52 501 839</td>
<td>52 501 839</td>
</tr>
<tr>
<td>R1L2</td>
<td>4 204</td>
<td>7 564</td>
<td>47 836 757</td>
<td>49 690 834</td>
<td>49 690 834</td>
</tr>
<tr>
<td>R1L4</td>
<td>4 048</td>
<td>8 287</td>
<td>46 530 294</td>
<td>46 530 294</td>
<td>46 530 294</td>
</tr>
<tr>
<td>R1L4</td>
<td>7 660</td>
<td>13 174</td>
<td>42 948 107</td>
<td>42 429 329</td>
<td>42 429 329</td>
</tr>
<tr>
<td>R1L4</td>
<td>4 516</td>
<td>9 146</td>
<td>53 299 847</td>
<td>55 161 562</td>
<td>55 161 562</td>
</tr>
<tr>
<td>R1L4</td>
<td>4 452</td>
<td>9 252</td>
<td>53 441 353</td>
<td>56 730 879</td>
<td>56 730 879</td>
</tr>
<tr>
<td>R1L4</td>
<td>5 724</td>
<td>11 170</td>
<td>49 707 212</td>
<td>49 951 444</td>
<td>49 951 444</td>
</tr>
<tr>
<td>R1L4</td>
<td>8 180</td>
<td>15 638</td>
<td>40 597 536</td>
<td>45 507 576</td>
<td>45 507 576</td>
</tr>
<tr>
<td>R1L4</td>
<td>4 032</td>
<td>10 263</td>
<td>59 225 243</td>
<td>61 240 482</td>
<td>61 240 482</td>
</tr>
<tr>
<td>R1L4</td>
<td>5 048</td>
<td>10 751</td>
<td>59 292 152</td>
<td>61 045 967</td>
<td>61 045 967</td>
</tr>
<tr>
<td>R1L4</td>
<td>6 368</td>
<td>13 219</td>
<td>54 975 374</td>
<td>55 514 383</td>
<td>55 514 383</td>
</tr>
<tr>
<td>R1L4</td>
<td>8 394</td>
<td>17 755</td>
<td>40 608 497</td>
<td>47 025 795</td>
<td>47 025 795</td>
</tr>
<tr>
<td>R1L4</td>
<td>2 660</td>
<td>7 986</td>
<td>7 440 845</td>
<td>7 387 963</td>
<td>7 387 963</td>
</tr>
<tr>
<td>R1L4</td>
<td>3 044</td>
<td>9 311</td>
<td>7 983 383</td>
<td>7 826 762</td>
<td>7 826 762</td>
</tr>
<tr>
<td>R1L4</td>
<td>3 616</td>
<td>13 502</td>
<td>9 435 913</td>
<td>7 359 779</td>
<td>7 359 779</td>
</tr>
</tbody>
</table>

page, as researchers all over the world find new solutions.

Table 2 is adapted from PESPlib’s website and signals in bold our results whenever they outperformed the previously best ones. The results are ordered by objective value (lower is better). We were able to actually improve the previously known best values on some problems, as is shown in that table.

The results achieved by our machine learning based approach surpassed the ones obtained with other state-of-the-art approaches in several problems, namely the R1L3, R2L1, R2L4, BL1, BL2, BL3, and BL4. Moreover, for the latter four mentioned problems, our results remain (as of 19th February 2018) at the top of the table, which allows us to say that our approach is currently positioned amongst the state-of-the-art approaches for solving this kind of problems.

Since the results published on the PESPlib web page do not feature running times, we were not particularly worried about them. Therefore, we did not impose a hard time limit on the execution time for this experiment, and let the algorithm run until we were satisfied with the results. We ended up giving several days of execution time to most of the instances. Nevertheless, to give an impression on the performance of our approach we can say that, for most of the instances, we could achieve in just about eight to twenty hours results within 2.5%-10% from the ones on Table 2. That is, we could achieve results that lie very close to the ones on Table 2 in less than a day, which is acceptable for long-term scheduling.

4.3 Comparison between the binary search and machine learning approaches

We would like to apply both our binary search and ML approaches to the set of PESPlib problems to compare their performance, but unfortunately we cannot encode all the pseudo-boolean constraints of any single instance from that library for our binary search approach since it would not fit in the 64GB of memory that we had available. Therefore, the first conclusion is straightforward: many real-world, large-sized (actually, interesting) problems are simply out of the question for the binary search approach.

However, if we discard the binary search itself, but keep the best solution found by the core guided heuristic used on that approach, we can actually compare it with the best solution found by the ML approach. Figure 4 shows precisely the results of that comparison.

As we can see, the ML approach could find better solutions than the ones found by our core guided heuristic on all PESPlib’s problems with no exception, with improvements ranging 21% to 33%. In fact, our best solutions for the PESPlib problems were found by the ML approach (see Section 4.2). Therefore, we realised that a machine-learned heuristic, developed by trial and error from the data underlying the problem being solved, outperformed our human-designed core guided heuristic, which makes use of general knowledge from the problem domain, at least on large problems.

In order to compare the actual core guided + binary search approach with the ML approach, we had to artificially develop some problems whose encoding, when approached by the binary search, could fit into 64GB.
We started with the real-world problem $M1_{24}$ from Table 1 and, in order to harden the problem, progressively added to it artificial connection constraints simulating the interest of some passengers in transferring from one train to another at some intersection stations. Then we did the same on problem $M6_{120}$, and ended up producing the instances presented on Table 3.

We have set a time limit of two hours and collected, for both approaches, the number of iterations that the algorithm could perform, and the value of the best solution found with the travel time objective function. In the case of the binary search approach, we separated the solution values and number of iterations for the core guided heuristic (which runs first) from the ones of the binary search procedure itself (which gives the final solution).

In this set of problems the picture is very different from what we saw for the PESPlib. The ML approach does not outperform the human-designed approach on all instances anymore. In fact, the solutions found by the binary search approach are better and produced in less time in most of the cases. There is no much surprise in these results, since the binary search approach is an exact method, so it is expected to produce the optimal solution, if it is given sufficient time, or at least optimised solutions which are better than an approximate method such as our ML approach. However, what is interesting to note is that the core guided heuristic itself finds solutions that are pretty close to the final best solution found by the binary search procedure. In fact, on the instance $M1_{24\_original}$, the core guided heuristic actually found the optimal solution and thus the binary search procedure was completely bypassed\(^\text{13}\). Therefore, the results indicate that our core guided heuristic combined with the binary search is still better than our ML approach on smaller problems, whereas the latter approach unleashes its full potential on bigger problems (such as the PESPlib) where the former soon gets stuck.

In light of the above, we concluded that the binary search approach is well-suited for small problems and its greatest advantage is the capability of finding the optimal solution and proving its optimality\(^\text{14}\). However, it suffers from the curse of dimensionality, and so consumes unacceptable amounts of memory and becomes impractical on real-world, large-sized problems. If we discard the binary search process and use just the core guided heuristic to produce solutions (thus avoiding the memory problems, but also loosing optimality guarantees), we can address bigger problems, but we cannot go as far as we can with the ML approach under the same conditions, in general.

The ML approach, on the other hand, is capable of approaching even the largest instances and find good,
optimised solutions, even outperforming other state-of-the-art solvers (Section 4.2). Nonetheless, it cannot guarantee the optimality of the found solutions, not to mention the fact that it is not a “well-behaved” algorithm in the sense that it never really ends — actually we impose a time limit and content ourselves with the best solution found up to that time.

5 Conclusion

We proposed two approaches for solving PTOPs, both relying, in their core, on a SAT solver. One is an exact method, consisting in a binary search procedure in the range of possible solution values supported by incremental SAT calls and an heuristic to compute better upper bounds. The other one is a reinforcement learning, multi-agent approach capable of addressing bigger problems by automatically exploring and exploiting seemingly promising combinations of process times.

While several authors have proposed both SAT and MaxSAT approaches to solve PTSPs and PTOPs before [Großmann et al (2015); Gattemann and Nachtigall (2016)], the current state-of-the-art for optimising timetables with respect to the total travel time is still based on Mixed Integer Programming (MIP) approaches [Goerigk and Liebchen (2017)], namely on the modulo simplex [Nachtigall and Opitz (2008a,b)]. Because we believed that the potential of SAT techniques was not fully explored yet in this domain, we decided to contribute to this family of techniques, having achieved results — namely on the PESPlib problems — that lie close to those obtained by other state-of-the-art approaches. In fact, we were able to actually submit new records on seven of the twenty PESPlib instances.

The results also suggest that our proposed SAT-based approaches may outperform existing equivalent MaxSAT, MILP and CP approaches, at least on a set of real-world problems that we had access to. This improvement is felt, on some problems, in terms of the computational time needed to solve them, and on some other problems in terms of the quality (i.e., the total travel time, for instance) of the generated timetable, or both. Moreover, the comparison analysis presented in this paper showed that our binary search based approach, while capable of solving several problems to proven optimality, is limited in the size of the problems it can address, due mainly to memory constraints. Our multi-agent, reinforcement learning approach, in contrast, exhibited the ability to address those limitations and thus cope with larger problem instances.

Apart from the published results, our work innovates by combining for the first time (to our knowledge) SAT and reinforcement learning techniques to solve a PTOP. While the techniques that we employed in our reinforcement learning approach were very simple and by no means new nor the state-of-the-art in the field of ML, they could successfully address our biggest problems and produce better solutions than those achieved by our “hand-designed” heuristic and binary search procedure. Therefore, we hope that our results encourage other future developments in the field of PTOPs using ML, which seems promising to us. If one conclusion can be derived from this whole research, which started with the assumption that the potential of SAT in the field of PTOPs was still not fully unleashed, it is that the potential of combined ML and SAT techniques in this field is still in its beginnings and is already providing up-and-coming results.

Another not so obvious accomplishment of this work is the fact that we successfully implemented a solution for optimising timetables in SISCOG’s timetabling product, ONTIME, that is economically more affordable than previous solutions based on expensive MIP solvers (like CPLEX), since it only requires a free and open-sourced SAT solver for its core engine. Furthermore, it also takes advantage from any improvements in the field of SAT, since when a new SAT solver is released we can simply replace our old SAT solver by the new one and instantly benefit from new state-of-the-art techniques and increased performance without changing a single line of code in our algorithms.
References


