Reprobate at ITC 2021
Pseudoboolean Optimisation for RobinX Sports Timetabling

Martin Mariusz Lester

Abstract We report on the development of Reprobate, a tool for solving sports timetabling problems in the RobinX format. The main approach used by the tool is to encode a timetabling problem using pseudoboolean (PB) constraints and solve it using existing solvers. Initially, it uses a monolithic encoding that attempts to satisfy all constraints simultaneously. If this finds a feasible solution, the tool can improve it using a separate encoding that tunes only the home/away pattern while fixing the pairings of teams. Furthermore, Reprobate employs a small portfolio of different solvers and encoding variations and returns the best solution found by any of them.

We entered Reprobate in the International Timetabling Competition 2021. It was able to find feasible solutions for the majority of instances, although it struggled to handle large break constraints. For those instances where it could initially find a solution, the combination of tuning, use of a portfolio of solvers, and variations in encoding yielded an average reduction in solution cost of 23%.

Keywords pseudoboolean constraints · sports timetabling

1 Introduction

The 2021 edition of the International Timetabling Competition (ITC) was based around solving sports timetabling problems presented in a restricted version of the RobinX format [5]. The competition was limited to time-constrained double round-robin (2RR) tournaments. In such a tournament, each of $n$ teams plays every other team exactly twice over $2(n - 1)$ slots: once at home and once away. The RobinX format allows a wide range of other constraints to be specified, for example concerning the slots in which particular matches may occur, or limiting the number of breaks where a team has consecutive home or consecutive away games [3]. Some
of these constraints are hard, meaning that they must be satisfied. Others are soft, meaning that they may be violated, but there is a cost for doing so. The goal is to find a solution that minimises the sum of costs of violated constraints. See the competition report for a full description [3].

We developed the tool Reprobrate for the competition. In Section 2 we outline the approach used by the tool and some of the optimisations we developed. We discuss its performance in Section 3 before concluding in Section 4.

Reprobate is implemented as a series of Perl scripts that invoke existing pseudo-boolean (PB) solvers such as clasp [6] and Sat4J [1]. The tool [8] and solvers are freely online under an open source licence.

2 Approach

At its core, Reprobrate extracts the constraints from a RobinX timetabling problem and encodes them monolithically as a pseudo-boolean (PB) optimisation problem, specifically a Weighted Boolean Optimisation (WBO) problem. Then, it uses an existing PB solver to solve the problem. If the solver is successful, Reprobrate extracts the timetable from the solution. To make the system more competitive, we implemented three optimisations: a portfolio of solvers, a tuning process, and some variations on the encoding.

2.1 Pseudo-boolean Constraints

The pseudo-boolean constraint satisfaction (PBS) [10] problem is a generalisation of the well-known boolean satisfiability (SAT) problem that makes it easy to express cardinality constraints. In a weighted boolean optimisation (WBO) problem, these constraints can be given costs, with the goal being to minimise the sum of costs. SAT-based approaches to sports timetabling have been considered before [11,7], but are relatively uncommon.

Our encoding uses the following sets of boolean variables:
1. $M_{t_1,t_2,s}$ — true just if team $t_1$ plays home against team $t_2$ in slot $s$;
2. $H_{t,s}$ — true just if team $t$ plays home in slot $s$;
3. $B_{t,s,h}$ — true if team $t$ has a home break ($h = 0$)/an away break ($h = 1$) in slot $s$, with $s > 0$.

The timetable is determined by the $M$ variables; the remaining auxiliary variables make it easier to express certain constraints.

Linear PB constraints are equivalent to 0-1 Integer Linear Programming (0-1 ILP), which is itself a restriction of Mixed Integer Programming (MIP). However, there is an important practical difference between PB solvers and 0-1 ILP solvers. PB solvers tend to use techniques from SAT solvers, such as clause-driven conflict learning (CDCL). In contrast, 0-1 ILP solvers tend to use techniques from linear programming (LP). According to Berthold and others [2], “feasibility problems with many constraints that have 0/1 coefficients only” tend to work best with PB solvers, but “instances with many inequalities with arbitrary coefficients” tend to work best with MIP solvers. Our encoding uses only +/- 1 coefficients, so all constraints are either pure SAT constraints or cardinality constraints. We investigated
the possibility of using commercial MIP solvers such as Gurobi and CPLEX with our encoding, but found that they performed very poorly.

2.2 Portfolio of Solvers

It is well-known that different SAT solvers perform well on different SAT instances. Therefore, if one wishes to solve a particular instance, it is most effective to run several different solvers in parallel and see which one (if any) finishes first. This portfolio approach is used by the most competitive SAT solvers, although it is banned from the main track of the SAT competition, as portfolio solvers tend not to contribute towards the development of new techniques. The same is true for PB instances, so in Reprobate, we use 2 solvers (clasp and Sat4J) with a range of options.

2.3 Portfolio of Encodings

SAT and PB solvers are often sensitive to the exact encoding of the constraints in a problem. For this reason, we implemented 6 variations on our initial encoding, configurable using command-line switches. As the number of variations is small, by default Reprobate simply tries all of them individually and picks the best solution found by any of them.

2.4 Tuning Process

Many PB solvers are complete, in the sense that they are guaranteed eventually to find an optimal solution if one exists. However, in practice, this often takes infeasibly long, and even if a solver finds a feasible solution, it may not be optimal. In previous work on generating tournament timetables for the 4-player game mahjong [9], we found that we could often improve a timetable using a separate encoding that fixes the groupings of players in a particular round. It is possible that the optimal solution to the original problem is no longer a solution to this modified encoding, but in practice this is not a problem, as it always produces at least as good a solution as we could find without it.

Reprobate uses a similar technique. After it has found a feasible solution, it generates a separate encoding of the timetabling problem in which pairings of teams in each slot are fixed, but not the choice of which team plays home and which team plays away. Again, it solves this using a PB solver and extracts a timetable from the solution. If the tuned timetable is an improvement on the best timetable produced by the monolithic encoding, Reprobate returns that; otherwise, it returns the original solution.

3 Results

Using the default encoding and the best solver from our portfolio (clasp with the crafty preset) with a timeout of 600 s, Reprobate was able to find feasible
solutions for 25 out of 45 problem instances (56%) in the ITC 2021. Table 1 shows the corresponding objective scores, which we adopt as our baseline. Applying the portfolio of solvers and encodings during the ITC 2021, we increased this to 29 out of 45 (64%). The addition of another encoding variation after the competition [7] increased this to 33 (73%).

This was not competitive in the ITC 2021, where it did not place in the top half of the results. However, according to the competition report, “for most problem instances, a straightforward integer programming formulation could not even generate a feasible solution”, so it is better than that. Of the instances Reprobate could not solve during the competition, all except Middle 3 contained a large, hard BR2 constraint that limited the number of breaks permitted in the timetable.

Focusing now just on the 25 instances in our baseline, Figure 1 shows the relative improvement made by our optimisations, as well as the best scores submitted during the competition; Table 2 shows the raw numbers. In combination, the optimisations we made yielded an average decrease in objective of 23%, although this is still some way off the best solutions found for most instances.

4 Conclusion

We have developed Reprobate, the first PB-based tool for solving RobinX sports timetabling problems. To the best of our knowledge, this is the first general-purpose sports timetabling tool that uses the OPB file format (for PBS/WBO problems) and its associated solvers. While Reprobate is effective for many timetabling problems, it struggles to handle large break constraints. This is a known limitation of SAT-based approaches, for which he have implemented some existing mitigations, but more work is needed to investigate how best to handle these.

References

Fig. 1 Effect of tuning on objective, for the baseline, the portfolio and different switches. ITC best solutions included for comparison.

Table 2 Effect of tuning on objective, for the baseline, the portfolio and different switches. ITC best solutions included for comparison.