Generating Hard Satisfiable Scheduling Instances

Josep Argelich\textsuperscript{1}, Ramón Béjar\textsuperscript{2}, Alba Cabisco\textsuperscript{1}, César Fernández\textsuperscript{1},
Felip Manyà\textsuperscript{1}, and Carla P. Gomes\textsuperscript{2}

\textsuperscript{1} Departament d’Informàtica i Enginyeria Industrial, Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
\{alba,cesar,felip\}@eup.udl.es
\textsuperscript{2} Department of Computer Science, Cornell University
Ithaca, NY 14853, USA
\{bejar,gomes\}@cs.cornell.edu

Abstract. We present a random generator of partially complete round robin timetables that produces exclusively satisfiable instances, and provide experimental evidence that there is an easy-hard-easy pattern in the computational difficulty of completing partially complete timetables as the ratio of the number of removed entries to the total number of entries of the timetable is varied. Timetables in the hard region provide a suitable test-bed for evaluating and fine-tuning local search algorithms.

1 Introduction

Local search algorithms (LSA’s) are widely used to efficiently solve planning and scheduling problems [3, 9]. One difficulty with LSA’s is that they are incomplete and cannot prove unsatisfiability. Thus, benchmark instances for measuring the performance of LSA’s have to be satisfiable. Unfortunately, it has proven to be surprisingly difficult to develop random generators of hard satisfiable instances of combinatorial problems [1].

Given a set of candidate benchmark instances, unsatisfiable instances are generally filtered out with complete algorithms, and then only satisfiable instances are used to evaluate and fine-tune LSA’s. However, this approach is problematic in problems where incomplete algorithms can solve larger instances than complete algorithms because the latter cannot identify hard satisfiable instances.

In this paper we describe a random generator of satisfiable scheduling instances which are computationally difficult to solve with LSA’s for SAT. Our generator starts by randomly creating a timetable $T$ for a temporally dense single round robin tournament using the incomplete satisfiability solver WalkSAT [11]. Then, it generates a partially complete round robin timetable $T'$ by randomly removing a given number of entries of $T$ in such a way that the number of removed entries in each column and each row are approximately equal. The underlying generation model guarantees that $T'$ can be completed into a feasible timetable, and has the advantage that the expected hardness of completing a
partially complete timetable can be finely controlled by tuning the number of removed entries.

In order to investigate the hardness of the instances of our generator we conducted a comprehensive experimental investigation. We observed that there is an easy-hard-easy pattern in the computational difficulty of completing partially complete timetables with LSA’s for SAT as the ratio of the number of removed entries to the total number of entries is varied. Timetables in the hard region provide a suitable set of randomly generated satisfiable scheduling benchmarks.

We considered the generic problem solving approach that consists in modeling combinatorial problems as SAT instances and then solving the resulting instance with a SAT solver. In the last years the planning as satisfiability approach has gained popularity and has allowed the creation of planning systems like Blackbox [9]. The scheduling as satisfiability approach was used by Crawford & Baker [5] to solve the job shop problem and by Béjar & Manyà [2, 3] to create timetables for a variant of round robin tournaments. The generation of satisfiable instances for the quasigroup completion problem was investigated by Achlioptas, Gomes, Kautz & Selman [1, 6], as well as in [8]. Their papers inspired our work on the round robin completion problem.

The paper is structured as follows. In Section 2 we introduce the round robin problem. In Section 3 we describe the random generator of partially complete timetables. In Section 4 we present and discuss the experimental investigation.

2 The round robin problem

In this paper we consider the timetting problem for temporally dense single round robin tournaments (DSRR): given an even number of teams \( n \), the DSRR problem consists in distributing \( n(n - 1)/2 \) matches in \( n - 1 \) rounds in such a way that each team plays each other team exactly once during the competition. Figure 1 shows a 6-team DSRR timetable. We represent DSRR timetables for \( n \) teams by an \( n \times (n - 1) \) matrix \( o \) of variables, where variables \( o_{t,r} \) tell the opponent team against which team \( t \) plays in round \( r \).

The DSRR problem for \( n \) teams can be represented as a constraint satisfaction problem (CSP) [7] as follows:

- The set of variables is formed by all variables \( o_{t,r} \) in matrix \( o \).
- The domain \( D_{o_{t,r}} \) of each variable \( o_{t,r} \) is \( \{1, \ldots, n\} \).
- The set of constraints is formed by the following constraints:
  - all-different\((o_{t,1}, \ldots, o_{t,n-1})\), for every \( t \in \{1, \ldots, n\} \), and
  - round-matches\((o_{1,r}, \ldots, o_{n,r})\), for every \( r \in \{1, \ldots, n - 1\} \).

The constraints all-different and round-matches are defined as follows:

\[
\text{all-different}(x_1, \ldots, x_m) = \{(v_1, \ldots, v_m) \in D_{x_1} \times \cdots \times D_{x_m} | \forall i, j, i \neq j \ v_i \neq v_j\}
\]

\[
\text{round-matches}(x_1, \ldots, x_m) = \{(v_1, \ldots, v_m) \in D_{x_1} \times \cdots \times D_{x_m} | \forall i, j, i \neq j \ v_i \neq i \land v_i = j \leftrightarrow v_j = i\}
\]
The all-different constraint expresses that each row of a DSRR timetable contains every team only once, and the round-matches constraint expresses that each column groups the teams into matches; each column represents all the matches of one round.

Next, we define the SAT encoding of the n-team DSRR problem that we used in the experimental investigation described in Section 4.

1. The set \(\{p_{i,j}^k|1 \leq i \leq n, 1 \leq j \leq n - 1, 1 \leq k \leq n, i \neq k\}\) is the set of propositional variables. The intended meaning of \(p_{i,j}^k\) is that team \(i\) plays against team \(k\) in round \(j\).
2. The constraint all-different\((p_{1,1}, \ldots, p_{n,n-1})\) is defined as follows:

\[
\bigwedge_{1 \leq j < n} \left( \bigvee_{1 \leq k \leq n, k \neq i} p_{i,j}^k \right) \wedge \bigwedge_{1 \leq k_1 < k_2 \leq n, k_1 \neq k_2 \neq i} \left( \neg p_{i,j}^{k_1} \vee \neg p_{i,j}^{k_2} \right) \wedge \bigwedge_{1 \leq j_1 < j_2 < n} \left( \bigwedge_{1 \leq k \leq n, k \neq i} \neg p_{j_1,j_2}^k \vee \neg p_{j_1,j_2}^k \right)
\]

3. The constraint round-matches\((p_{1,1}, \ldots, p_{n,n})\) is defined as follows:

\[
\bigwedge_{1 \leq i \leq n} \left( \bigwedge_{1 \leq k \leq n, k \neq i} \left( \neg p_{i,j}^k \vee p_{i,j}^k \right) \right)
\]

We define the DSRR completion problem to be the problem of determining whether a partially complete DSRR timetable can be completed into a feasible timetable. In Section 4 we provide experimental evidence that completing a DSRR timetable is computationally harder than constructing a full timetable.

The DSRR completion problem is NP-complete. This follows from the fact that it is equivalent to the problem of completing partially complete symmetric quasigroups, which is known to be NP-complete [4]. Figure 2 shows the symmetric quasigroup of size 6 that corresponds to the DSRR timetable of Figure 1. We use the symbols \(\{1, 2, 3, 4, 5, X\}\) to fill the entries of the quasigroup the entry in row \(i\) and column \(j\) is \(r \in \{1, 2, 3, 4, 5\}\) if team \(i\) plays against team \(j\) in round \(r\), and the entries of the diagonal of the quasigroup are \(X\).
3 A random generator of partially complete timetables

The random generator of partially complete DSRR timetables that we have designed and implemented has two peculiarities: (i) produces exclusively satisfiable instances, and (ii) the number of removed entries in each column and each row are approximately equal. It has been shown recently that removing entries in a balanced way allows one to generate hard quasigroup completion problems [8].

The pseudo-code is shown in Figure 3: we represent entries by \( o'_{t,r} \) and refer to removed entries as \textit{holes}. The intended meaning of \( o'_{t,r} \) is that team \( t \) plays against team \( t' \) in round \( r \).

\begin{verbatim}
procedure Random-Generator
input: an even number of teams \( n \) and an even number of holes \( h \)
output: an \( n \)-team partially complete timetable with \( h \) holes

\[
h' := \lceil \frac{h}{n-1} \rceil + 1
\]
\( T := \) a randomly generated \( n \)-team complete DSRR timetable

while \( h > 0 \) do

\( S := \) set of non-empty entries \( o'_{t,r} \) of \( T \) such that rows \( t, t' \) have

\( \) have less than \( h' + 1 \) holes and column \( r \) has less than \( h' \) holes;

\( o'_{t,r} := \) a randomly selected entry of \( S \);

\( T := T \) with entries \( o'_{t,r} \) and \( o'_{t',r} \) removed;

\( h := h - 2; \)

endwhile

return(\( T \));

Fig. 3. Random generator of partially complete timetables
\end{verbatim}

In the experimental investigation, we used WalkSAT and the SAT encoding defined in Section 2 to randomly generate a complete DSRR timetable. SAT encodings of partially complete timetables were obtained by adding the list of holes to the SAT encoding of the corresponding complete timetable. As the resulting encodings had a considerable number of unit clauses, they were first simplified by applying unit propagation and then solved with WalkSAT.

It is worth mentioning that WalkSAT takes less than 1 minute to find a complete timetable for 40 teams. Systematic satisfiability algorithms like Satz [10] are not able to solve complete DSRR timetable for 14 teams after 48 hours.

4 Experimental results

In the experimental investigation we first used the random generator of partially complete timetables to produce sets of instances for different number of teams: \( n = 30, 32, 36 \); we considered these values of \( n \) in order to get experimental results in a reasonable amount of time. For all the sets, we varied the ratio of the number of holes \( h \) to the total number of entries of the timetable \( (n \times (n - 1)) \) from 0.350 to 0.420 for \( n = 30, 32 \), and from 0.320 to 0.400 for \( n = 36 \); we incremented that ratio by 0.001 in each step. At each setting we ran WalkSAT on 20 partially
complete timetables. Each instance was executed until 25 solutions were found using no cutoff (maxflips), 30% noise for $n = 30$, 26% noise for $n = 32$, and 20% noise for $n = 36$. We used approximately optimal noise parameter settings for each timetable size. Such experiments were performed on PC’s with 500 MHz Pentium III Processors under Linux Operating System.

Figure 4 visualizes the easy-hard-easy pattern in the computational difficulty of completing partially complete timetables with WalkSAT for $n = 30, 32, 36$. Along the horizontal axis is the ratio of the number of holes to the total number of entries of the timetable, and along the vertical axis is the median number of flips needed to solve an instance.

Figure 5 is like Figure 4 but the median number of flips are normalized. One can observe that there is a shift in the location of the hardness peak as a function of the number of teams.

Figure 6 visualizes the easy-hard-easy pattern for $n = 32$ showing seconds instead of flips. Along the vertical axis are the mean and median number of seconds needed to solve an instance.

Fig. 4. Computational cost profiles Fig. 5. Normalized computational cost profiles for 30, 32 and 36 teams
From the experimental results we can conclude that, when we use our random
generator of partially complete timetables, there is an easy-hard-easy pattern in
the computational difficulty of completing partially complete timetables with
LSA’s for SAT as the ratio of the number of removed entries to the total number
of entries is varied. Thus, the expected hardness of completing a timetable can
be finely controlled by tuning the number of removed entries, and timetables in
the hard region provide a source of suitable scheduling benchmarks to evaluate
and fine-tune LSA’s.

Taking into account the existing work on the quasigroup completion prob-
lem [1,8], and the equivalence between the DSRR completion problem and the
problem of completing partially complete symmetric quasigroups, we conjecture
that the easy-hard-easy pattern could also be observed if we use non-Boolean en-
codings as well as algorithms other than WalkSAT. A crucial point for obtaining
this difficulty profile was the generation model of partially complete timetables.

Acknowledgements: Research partially funded by the DARPA contracts
F30602-00-2-030 and F30602-00-2-0396, and project CICYT TIC96-1038-C04-03.
The fifth author was supported by the “Secretaría de Estado de Educación y
Universidades”. This research was also partially funded by the Intelligent
Information Systems Institute, Cornell University, funded by AFRL/AFOSR
(F49620-01-1).

References

2. R. Béjar and F. Manyà. Solving combinatorial problems with regular local search
4. C. Colbourn. Embedding partial steiner triple systems is NP-complete. Journal
5. J. M. Crawford and A. B. Baker. Experimental results on the application of
7. M. Henz, T. Müller, S. Thiel, and M. van Brandenburg. Global constraints for
Balance and filtering in structured satisfiable problems. In Proc. of IJCAI’01,