

The Round Robin Problem

In this section we first introduce the round robin problem and then define the reduction to classical propositional logic that we used in our experiments. In the below description of the round robin problem we follow closely the presentation of Gomes *et al.* (1998).

In sports scheduling one of the issues is to find a feasible schedule for a sports league that takes into consideration constraints on how the competing teams can be paired, as well as how each team's games are distributed in the entire schedule. Here we consider the timetabling problem for the classic "round robin" schedule: every team must play every other team exactly once. The global nature of the pairing constraints makes this a particularly hard combinatorial search problem.

A game will be scheduled on a certain field at a certain time. This kind of combination will be called a slot. These slots can vary in desirability due to such factors as lateness in the day, the location and the condition of the field, etc. The problem is to schedule the games such that the different fields are assigned to the teams in an equitable manner over the course of the season.

The round robin problem for n teams (n -team round robin problem) is formally defined as follows:

1. There are n teams (n even) and every two teams play each other exactly once.
2. The season lasts $n - 1$ weeks.
3. Every team plays one game in each week of the season.
4. There are $n/2$ fields and, each week, every field is scheduled for one game.
5. No team plays more than twice in the same field over the course of the season.

The meeting between two teams is called a game and takes place in a slot; i.e., in a particular field in a particular week. Table 1 shows a solution for the 10-team round robin problem; teams are named 1, . . . , 10. An n -team round robin timetable contains $n(n - 1)/2$ slots and slots are filled in with games. A game is represented by a pair of teams (t_1, t_2) such that $t_1 < n$ and $t_1 < t_2$.

The combinatorics of the round robin problem are explosive (McAloon, Tretkoff, & Wetzel 1997): For an n -team league, there are $n/2 \cdot (n - 1)$ games (i, j) with $1 \leq i < j \leq n$ to be played. A schedule can be thought of as a permutation of these games. So, for n teams the search space size is $(n/2 \cdot (n - 1))!$; i.e., the search space size grows as the factorial of the square of $n/2$.

The n -team round robin problem is encoded as an instance of the SAT problem as follows:

1. The set of propositional variables is

$$\{p_{ij}^{1k} \mid 1 \leq i \leq n/2, 1 \leq j, k \leq n - 1\} \cup \\ \{p_{ij}^{2k} \mid 1 \leq i \leq n/2, 1 \leq j \leq n - 1, 2 \leq k \leq n\}$$

and its cardinality is $n(n - 1)^2$.

Each slot in the timetable is filled in by a pair of variables $(p_{ij}^{1k_1}, p_{ij}^{2k_2})$. The intended meaning of the pair

$(p_{ij}^{1k_1}, p_{ij}^{2k_2})$ is that team k_1 will play against team k_2 in field i in week j .

2. In each slot, one team plays against another team. For each slot $(p_{ij}^{1k_1}, p_{ij}^{2k_2})$ ($1 \leq k_1 \leq n - 1, 2 \leq k_2 \leq n$), we define the clauses

$$(p_{ij}^{11} \vee \dots \vee p_{ij}^{1n-1}) \wedge (p_{ij}^{22} \vee \dots \vee p_{ij}^{2n})$$

These clauses together with the clauses in (4) ensure that one team plays exactly against another team every week.

3. In each slot $(p_{ij}^{1k}, p_{ij}^{2k'})$ it holds that $k < k'$. For each two teams k_1, k_2 such that $k_1 \geq k_2$, we define the clause

$$\neg p_{ij}^{1k_1} \vee \neg p_{ij}^{2k_2}$$

4. Every team plays one game in each week of the season. For each week j , for each team k , for each two fields i_1, i_2 ($1 \leq i_1, i_2 \leq n/2$) and for r_1, r_2 ($1 \leq r_1, r_2 \leq 2$), we define the clause

$$\neg p_{i_1 j}^{r_1 k} \vee \neg p_{i_2 j}^{r_2 k}$$

provided that $p_{i_1 j}^{r_1 k} \neq p_{i_2 j}^{r_2 k}$. Clauses containing a variable of the form p_{ij}^{21} or p_{ij}^{1n} are not generated.

5. Every two teams play each other exactly once. For each two different slots of the form $(p_{i_1 j_1}^{1k_1}, p_{i_1 j_1}^{2k_2})$ and $(p_{i_2 j_2}^{1k_1}, p_{i_2 j_2}^{2k_2})$ such that $j_1 \neq j_2$ and $k_1 < k_2$, we define the clause

$$\neg p_{i_1 j_1}^{1k_1} \vee \neg p_{i_1 j_1}^{2k_2} \vee \neg p_{i_2 j_2}^{1k_1} \vee \neg p_{i_2 j_2}^{2k_2}$$

The clauses of (5) ensure that every two teams play each other at most once over the course of the season. Since the total number of slots coincides with the total number of possible games, the above clauses not only ensure that each possible game appears at most in one slot, but exactly once.

6. No team plays more than twice in the same field over the course of the season. For each team k , for each field i , for each three different weeks j_1, j_2, j_3 and for each r_1, r_2, r_3 ($1 \leq r_1, r_2, r_3 \leq 2$), we define the clause

$$\neg p_{ij_1}^{r_1 k} \vee \neg p_{ij_2}^{r_2 k} \vee \neg p_{ij_3}^{r_3 k}$$

The number of clauses of the SAT instance obtained for the n -team round robin problem is in $\mathcal{O}(n^6)$. By employing additional variables, it is possible to obtain a SAT instance with a number of clauses which is in $\mathcal{O}(n^4)$. Unfortunately, that reduction is not so computationally competitive. This fact was also observed in (Béjar & Manyà 1999c).

We have performed experiments with five alternative SAT encodings, but the results obtained were rather worse. In the rest of the paper, we always refer to the above SAT encoding.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Field 1	(6, 9)	(4, 6)	(1, 8)	(4, 10)	(2, 8)	(7, 9)	(5, 7)	(1, 2)	(3, 5)
Field 2	(2, 3)	(1, 5)	(2, 4)	(1, 7)	(9, 10)	(8, 10)	(3, 6)	(4, 9)	(6, 8)
Field 3	(5, 10)	(2, 7)	(3, 9)	(5, 9)	(1, 3)	(1, 6)	(4, 8)	(6, 10)	(4, 7)
Field 4	(1, 4)	(8, 9)	(5, 6)	(3, 8)	(6, 7)	(2, 5)	(1, 10)	(3, 7)	(2, 10)
Field 5	(7, 8)	(3, 10)	(7, 10)	(2, 6)	(4, 5)	(3, 4)	(2, 9)	(5, 8)	(1, 9)

Table 1: A 10-team round robin timetable

procedure WalkSAT

input: a set of clauses S , maxTries, maxFlips
and a heuristic H
output: a satisfying truth assignment of S , if found

```

for  $i := 1$  to maxTries do
{
   $I :=$  a randomly generated truth assignment;
  for  $j := 1$  to maxFlips do
  {
    if  $I$  satisfies  $S$  then return  $I$ ;
     $c :=$  a randomly selected clause not satisfied by  $I$ ;
     $p :=$  a propositional variable in  $c$  selected using  $H$ ;
     $I := I$  with  $p$  flipped;
  }
}
return "no satisfying truth assignment found";

```

Figure 1: Procedure WalkSAT

Local Search Algorithms for SAT

The WalkSAT algorithm is shown in Figure 1. It starts from a randomly generated truth assignment, and repeatedly selects one of the clauses that is not satisfied by the current assignment. Then, it selects one of the variables in that clause using a heuristic and flips its truth assignment. The algorithm flips truth assignments until a satisfying truth assignment is found or until some predefined number of flips (maxFlips) is reached. This process is repeated as needed, up to a maximum of maxTries times. Most heuristics take a noise parameter ω ($\omega \in [0, 1]$) to escape from local optima.

In fact, WalkSAT is a family of algorithms (Selman, Kautz, & Cohen 1994; McAllester, Selman, & Kautz 1997). The difference between the algorithms lies in the heuristic used to select the variable to be flipped next. The heuristics considered in this paper are the following ones:

G+Tabu: We maintain a list of a fixed size t , called tabu list, that contains bindings (i.e., pairs of the form (p, b) , where p is a variable and b is either 0 or 1). Such bindings represent the last t flips performed by WalkSAT.

Let ω be the noise parameter. With probability ω , pick any variable of the clause selected by WalkSAT; otherwise, pick a variable p of the clause selected by WalkSAT that (i) if the value that assigns the current truth assignment to p is changed to b , the new truth assignment minimizes the total number of unsatisfied clauses and (ii) the binding (p, b) is not in the tabu list. If all the possible flips in the clause selected by WalkSAT are tabu, choose a different unsatisfied clause instead. If all the possible flips in all the

unsatisfied clauses are tabu, then the tabu list is ignored. In contrast to (Selman, Kautz, & Cohen 1994), our tabu list contains a list of bindings instead of a list of variables.

R-Novelty: This strategy sorts the variables of the clause selected by WalkSAT by the total number of clauses that are not satisfied if the variable is flipped, but breaking ties in favor of the least recently flipped variable. Consider the best (p_i) and second-best (p_j) variable under this sort, and let n be the difference in the objective function between p_i and p_j . If the best variable is not the most recently flipped variable in the clause, then select it. Otherwise, there are four cases:

1. When $\omega < 0.5$ and $n > 1$, pick p_i .
2. When $\omega < 0.5$ and $n = 1$, then with probability 2ω pick p_j ; otherwise, pick p_i .
3. When $\omega \geq 0.5$ and $n = 1$, pick p_j .
4. When $\omega \geq 0.5$ and $n > 1$, then with probability $2(\omega - 0.5)$ pick p_j ; otherwise, pick p_i .

Additionally, to inhibit loops, the variable to be flipped is picked at random from the selected clause every 100 flips.

In the following, when we write WalkSAT/G+Tabu we mean WalkSAT using heuristic G+Tabu, and when we write WalkSAT/R-Novelty we mean WalkSAT using heuristic R-Novelty.

Experimental Results

First of all, we have implemented a generator of SAT instances of the round robin problem. Then, we have executed the instances with both systematic and local search satisfiability algorithms. With systematic algorithms, we were only able to solve the round robin problem for 8 teams using both Satz (Li & Anbulagan 1997) and the randomized version of Satz (Gomes, Selman, & Kautz 1998), and we solved the round robin problem for 10 teams using REL-SAT (Bayardo & Schrag 1997).

With local search algorithms, we have executed SAT instances of the round robin problem, ranging from 12 to 20 teams, with WalkSAT/G+Tabu and WalkSAT/R-Novelty using both approximately optimal noise parameters and different cutoff values. Table 2 shows the number of clauses of the SAT instances, and the cutoff values (maxFlips), the lengths of the tabu list, the noise parameters and the average number of restarts that gave rise to the best running times in our experiments.

For 12 teams and 14 teams we performed 200 tries and used a very high cutoff value in order to get a solution in

Teams	Clauses	maxFlips	WalkSAT/G+Tabu			WalkSAT/R-Novelty	
			tabu list	ω	avg. tries	ω	avg. tries
12-team	$2 \cdot 10^5$	∞	7	0.233	1	0.09	1
14-team	$5 \cdot 10^5$	∞	8	0.19	1	0.05	1
16-team	$12 \cdot 10^5$	$3.5 \cdot 10^6$	10	0.184	3.6	0.045	1.65
18-team	$25 \cdot 10^5$	$5 \cdot 10^6$	10	0.175	6	0.0328	4.5
20-team	$47 \cdot 10^5$	$12 \cdot 10^6$	10	0.142	16	0.0222	10

Table 2: Number of clauses per instance, parameter settings and average number of tries needed to find a solution

Teams	Béjar&Manyà (1999)	Gomes et al. (1998)	WalkSAT/G+Tabu	WalkSAT/R-Novelty
12-team	6 min	< 0.22 min	0.6 min	0.27 min
14-team	1.30 hrs	4.17 min	4.08 min	1.74 min
16-team	2.35 hrs	1.4 hrs	0.78 hrs	0.28 hrs
18-team	*	\approx 22 hrs	3.67 hrs	1.98 hrs
20-team	*	*	20.48 hrs	12.73 hrs

Table 3: Comparison of experimental results for the n -team round robin problem

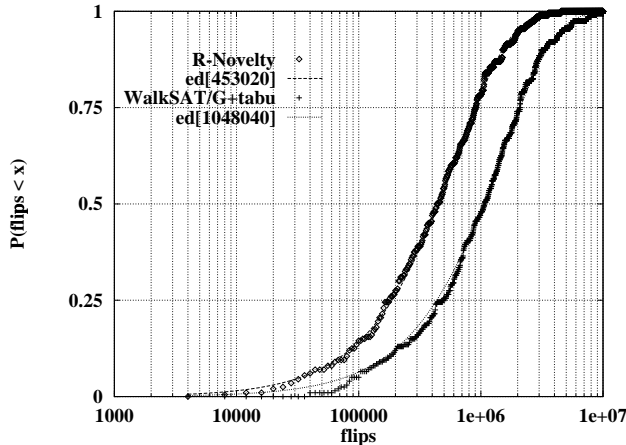


Figure 2: Run-length distributions for 14 teams

every try. This way, we obtained empirical run-length distributions (RLDs) for those instances, as well as the expected time and number of flips needed to solve an instance. Figure 2 shows the empirical RLDs obtained for the round robin problem for 14 teams using WalkSAT/G+Tabu and WalkSAT/R-Novelty. As can be seen in the figure, WalkSAT/R-Novelty outperforms WalkSAT/G+Tabu. Moreover, we found that the RLDs are well approximated by exponential distributions $ed[m]$, with distribution function $F(x) = 1 - 2^{-(x/m)}$, where m is the median of the distribution (see Figure 2). This kind of distributions were observed for local search algorithms in other problem domains by Hoos (1999) and Hoos & Stützle (1998).

For 16, 18 and 20 teams we executed WalkSAT/G+Tabu and WalkSAT/R-Novelty, with the input parameters shown in Table 2, until we found 10 solutions. In Table 3 are shown the average time needed to find a solution. This table also contains the times obtained in (Béjar & Manyà 1999c) and

in (Gomes *et al.* 1998) using other approaches.

Gomes *et al.* (1998) found a solution for 18 teams after 22 hours using a constraint programming formulation and a randomized constraint programming algorithm. Recently, we solved the round robin problem using a local search algorithm and an encoding based on many-valued propositional logic (Béjar & Manyà 1999c) (see (Béjar & Manyà 1999a; 1999b) for related work on the many-valued logic used to encode the round robin problem). We found solutions for 16 teams in about 2 hours; we believe that these results can be improved by introducing minor changes in the algorithm and using more suitable data structures for representing formulas.

Our experimental results show that, by combining local search satisfiability algorithms and an appropriate problem encoding based on classical propositional logic, we can find feasible schedules many times faster than using the best existing approaches to the round robin problem. We found solutions for 14 teams in less than 2 minutes, for 16 teams in less than 20 minutes and for 18 teams in less than 2 hours. Moreover, we found a solution for a previously unsolved instance (20 teams) in about 13 hours. These results provide experimental evidence that our approach scales better than previous approaches.

The application of the scheduling as satisfiability approach described in this paper, as well as the approach based on many-valued propositional logic that we proposed in (Béjar & Manyà 1999c), to a greater number of teams is the subject of further research. We expect that we will be able to find feasible schedules for the round robin problem for 22 teams in a reasonable time.

Acknowledgements

We thank T. Alsinet, C. Fernández, C. Mateu and F. Molina for technical assistance. This research was partially supported by the project SMASH (TIC96-1038-C04-03) funded

by the CICYT and “La Paeria”. The first author was supported by a doctoral fellowship of the Comissionat per a Universitats i Recerca (1998FI00326).

References

- Bayardo, R. J., and Schrag, R. C. 1997. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'97, Providence/RI, USA*, 203–208.
- Béjar, R., and Manyà, F. 1999a. A comparison of systematic and local search algorithms for regular CNF formulas. In *Proceedings of the 5th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'99, London, England*, 22–31. Springer LNAI 1638.
- Béjar, R., and Manyà, F. 1999b. Phase transitions in the regular random 3-SAT problem. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems, ISMIS'99, Warsaw, Poland*, 292–300. Springer LNAI 1609.
- Béjar, R., and Manyà, F. 1999c. Solving combinatorial problems with regular local search algorithms. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning, LPAR'99, Tbilisi, Republic of Georgia*, 33–43. Springer LNAI 1705.
- Crawford, J. M., and Baker, A. B. 1994. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the 12th National Conference on Artificial Intelligence, AAAI'94, Seattle/WA, USA*, 1092–1097.
- Gomes, C. P.; Selman, B.; McAloon, K.; and Tretkoff, C. 1998. Randomization in backtrack search: Exploiting heavy-tailed profiles for solving hard scheduling problems. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling, AIPS'98, Pittsburg/PA, USA*.
- Gomes, C. P.; Selman, B.; and Kautz, H. 1998. Boosting combinatorial search through randomization. In *Proceedings of the 15th National Conference on Artificial Intelligence, AAAI'98, Madison/WI, USA*, 431–437.
- Hoos, H. H., and Stützle, T. 1998. Evaluating las Vegas algorithms – pitfalls and remedies. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, UAI'98, San Francisco/CA, USA*, 238–245.
- Hoos, H. H. 1999. On the run-time behaviour of stochastic local search algorithms for SAT. In *Proceedings of the 16th National Conference on Artificial Intelligence, AAAI'99*, 661–666.
- Kautz, H. A., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'96, Portland/OR, USA*, 1194–1201.
- Kautz, H. A., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'99, Stockholm, Sweden*, 318–325.
- Li, C. M., and Anbulagan. 1997. Look-ahead versus look-back for satisfiability problems. In *Proceedings of the 3rd International Conference on Principles of Constraint Programming, CP'97, Linz, Austria*, 341–355. Springer LNCS 1330.
- McAllester, D.; Selman, B.; and Kautz, H. 1997. Evidence for invariants in local search. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'97, Providence/RI, USA*, 321–326.
- McAloon, K.; Tretkoff, C.; and Wetzel, G. 1997. Sports league scheduling. In *Proceedings of the 1997 ILOG Optimization Suite International Users' Conference, Paris, France*.
- Nemhauser, G. L., and Trick, M. A. 1998. Scheduling a major college basketball conference. *Operations Research* 46(1):1–8.
- Selman, B.; Kautz, H. A.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proceedings of the 12th National Conference on Artificial Intelligence, AAAI'94, Seattle/WA, USA*, 337–343.
- Warners, J. P., and van Maaren, H. 1999. A two phase algorithm for solving a class of hard satisfiability problems. *Operations Research Letters* 23(3–5):81–88.