Scheduling Social Tournaments

Iván Dotú

Departamento De Ingenier´ıa Inform´atica
Universidad Aut´onoma de Madrid

Abstract. The scheduling of social tournaments has attracted significant attention in recent years, as they arise in many practical applications and induce highly combinatorial problems. This research proposes a high-level modeling of social tournaments and a local search algorithm for finding solutions to the models. The effectiveness of the approach is demonstrated on three classes of applications: social golfer problems, debating tournaments, and judge assignements. In particular, the approach quickly solves real-life debating tournament and judge assignment problems that were open so far.

1 Introduction

Tournament scheduling problems arise in many practical applications and their highly symmetric and combinatorial nature makes them particularly challenging for search algorithms. It is thus not surprising that tournament scheduling has generated signifi cant attention from the constraint programming community (e.g.,[4, 10, 6, 8, 9, 2, 7]). In particular, much research in constraint programming has focused on the social golfer, which has become a standard benchmark; recent developments (e.g., [2, 7]) approach this problem using innovative, elegant, but also complex, symmetry-breaking schemes.

This research generalizes the modeling and local search approach proposed in [3] in order to schedule some challenging, real-life, social tournaments. Its main contribution is to show the versatility and effectiveness of this approach for a wide variety of social tournaments. In particular, the paper shows that

1. the approach schedules a challenging, open debating tournament and exhibits excellent performance across a wide range of such instances.

2. the approach fi nds the best known solution to a real-life, judge-scheduling problem which superimpose a referee assignment over a debating tournament.

The approach also schedules other challenging, social tournaments, which are briefly described in the conclusion but cannot be discussed here in detail for space reasons.

It is important to emphasize that these problems are not academic: social tournaments are increasingly pervasive and arise in very diverse settings. The applications described in this paper are also prototypical of many sport competitions. Moreover, they are extremely challenging computationally and have attracted signifi cant research in recent years. In particular, these applications have been tackled by a variety of algorithms and could not be solved so far. In addition, earlier local search algorithms (e.g., [1]) are not competitive with the approach presented here.

2 The Social Golfer

The first application is the well-known social golfer problem has attracted significant interest since its posting on `sci.op-research` in May 1998. The social golfer consists of scheduling $n = g \times p$ golfers into g groups of p players every week for w weeks so that no two golfers play in the same group more than once. An instance of the social golfer is specified by a triple $g - p - w$, where g is the number of groups, p is the size of a group, and w is the number of weeks in the schedule. We cannot develope the modeling of this problem here, but we refer the reader to [3] for an extended and complete explanation of both the modeling and the local search algorithm.

3 The Debating Tournament Problem

We now turn to the debating tournament problem (DTT), that can be seen as a generalization of the social golfer. The debating tournament problem is a real-life application given to us by W. Harvey [5]: "*A debating tournament involves 15 students. The students are grouped into 3 groups of 5 in each round, for 9 rounds. Due to the nature of the scoring system, it is desired to have as balanced a schedule as possible (i.e. each pair meets about the same number of times as every other pair). Having each pair meet 2 or 3 times is as balanced as you're going to get. I don't know whether this is possible.*" In the best known solution, obtained by constraint programming, every two students meet atmost four times. The local search algorithm presented here closes this DDT instance and finds a solution where every two students meet atmost three times. The experimental results also consider a variety of the DTTs which are generalizations of the social golfer problem, where golfers are allowed to meet up to k times.

The Modeling The DTP can be modeled like the SGP by relaxing the cardinality constraint, i.e., replacing the value "1" by k in the equations that define the constraints, the violations, and the neighborhood, in order to state not that two golfers cannot play in the same group more than "1" time, but that they cannot play more than k times in the same group, where $k = 2, 3, \ldots$

Experimental Results Tables 1 and 2 report the experimental results for SGLS for $k = 2, 3$ when the maximum number of moves was set to 100.000. Once again, the tables only give the results for those instances $g - p - w$ maximizing w since they also provide solutions for $w' < w$.

First observe that SGLS solves the real-life instance 3-5-9 for $k = 3$ in 0.1 second. This instance is in fact particularly challenging for systematic search, since constraint programming solutions were only able to find a solution with $k = 4$, demonstrating the effectiveness of SGLS [5].

The remaining results are not easy to comment since no other results are available at this point. It is useful however to make a few observations. A solution to the debating tournament problem for a given k could be obtained by solving the corresponding social golfer instance and repeating its schedule k times. Of course, such a solution is not really desirable in practice, since the exact same players will be meeting k times,

	size 3		size 4		size 5		size 6		size 7		size 8		size 9		size 10	
g	w	T	w	T	w	T	w	T	w	T	w	T	w	T	w	T
3	8	0	6	0	6	0.0	6	0.0	4	0	4	0.0	4	0.0	2	-
4	11	0.3	10	0.5	8	34.7 (90)	6	0.0	6	6.7	7	107.3 (60)	6	2.2	6	2.5
5	14	9.7 (19)	11	0.5	9	0.1	8	0.1	7	0.1	7	30.4 (4)	6	0.1	6	1.5
6	16	0.1	13	0.1	12	102.54 (99)	10	0.3	9	0.4	8	0.3	7	0.2	7	0.5
7	19	1	15	0.1	13	0.2	12	4.0	11	47.3	10	9.5	9	1.4	8	0.8
8	22	12.7 (2)	18	12.5 (1)	15	0.4	14	152.5 (23)	12	1.1	11	1.4	10	1.5	10	23.3
9	25	103.3 (86)	20	1.2	17	0.8	15	1.2	14	4.7	13	12.3	12	9.3	11	5.4
10	27	0.3	22	0.7	19	1.4	17	2.5	15	2.8	14	4.5	13	6.0	12	6.5

Table 1. CPU Times in Seconds for DDT(2) with the Maximal Number of Weeks.

	size 3		size 4		size 5		size 6		size 7		size 8		size 9		size 10	
g	w	T	w	T	w	T	w	T	w	T	w	T	w	T	w	T
3	12	0.0	11	2.1	9	0.1	9	0.1	9	0.5	9	0.4	9	0.4	7	30.3 (7)
4	16	0.1	14	6.0 (3)	12	0.1	11	0.2	10	0.1	9	0.1	9	0.5	8	0.1
5	21	5.7 (3)	17	0.1	15	0.1	14	0.6	13	1.3	12	0.6	11	0.4	11	8.5
6	25	5.0	21	2.0	18	0.2	17	2.1	16	61.4 (2)	15	107.6 (7)	14	7.8	13	1.9
7	29	0.4	25	144.2 (88)	22	12.3	20	7.7	18	1.3	17	2.3	16	2.8	15	3.0
8	33	0.2	28	0.8	25	2.8	23	19.4	21	4.0	20	27.1	19	101.9	18	36.5
9	38	82(19)	32	19.8	28	2.2	26	35.6	24	15.3	22	7.3	21	11.7	20	16.7
10	42	2.2	35	1.5	31	2.9	29	64.0	27	334.5 (3)	25	22.8	24	372.5 (1)	22	21.2

Table 2. CPU Times in Seconds for the DTP(3) with the Maximal Number of Weeks.

but it provides a lower bound $k \times w$ on the maximum number of weeks (where w is the maximal number of weeks for the social golfer problem). The experimental results depicted in the table indicate that SGLS improves this lower bound in almost every instance and matches it in the remaining ones. Moreover, the improvements become more signifi cant as the number of groups and the group sizes grow. It is also interesting to emphasize that most instances are solved really quickly (i.e., in less in a second in many cases), confi rming the results of the social golfer.

Finally, observe that instance 6-6-3 is optimal for the social golfer [9]. However, its special structure does not carry over to larger k, since SGLS fi nds schedules with up to 10 weeks ($k = 2$) and 17 weeks ($k = 3$).

4 The Judge Assignment Problem

This section considers the judge assignment problem, another real-life problem (also given to us by W. Harvey) [5]. The problem can be viewed as superimposing a judge assignment on top of a debating tournament instance. It was stated as follows: "*This problem involves the assignment of judges to the player schematic. Every player is required to furnish the tournament director with one judge. Each judge has to judge whichever sections in whichever rounds that the tournament director assigns. Every*

section is judged by exactly three judges. A judge can not judge two or more sections in the same round (because all three sections in a round are played simultaneously), and a judge can not judge any section in which his own player is playing. The tournament director wants to make sure that no judge judges any particular player more than twice over the nine rounds. Ideally, each judge should judge each player the same number of times (or something close to that). Given a particular player schematic, is it possible to get away with using only the fifteen judges provided by the players, and still meet the constraints? If not, the goal is to get close; the tournament director has extra judges available for spot duty, but it is better to use them as little as possible, for all kinds of good and valid reasons. The judge schematic is generated after the player schematic." The real-life instance consists of a 3-5-9 debating tournament and a judge assignment that assigns three judges to each group. The goal is to minimize the number of additional judges in the assignment, while satisfying the feasibility constraints.

The algorithm described in this section only considers the judge schematic. As specified, the player schematic is obtained first and is a debating tournament problem. The real-life problem superimposes the judge assignment on the debating tournament.

The Modeling The modeling is also similar to the SGP. It receives, as input, a solution to the debating tournament problem. In particular, $x[w, g, p]$ will denote the player scheduled in position p of group g in week w. The decision variables $y[w, g, p]$ associates a decision variable with every week, group, and position, where the set P_r of positions for the judges is included in the set of positions of the debating tournament (i.e., $P_r \subset P$). The goal is to find a schedule σ, where the value $\sigma(y[w, g, p])$ denotes the judge scheduled in position $\langle w, g, p\rangle$. In the following, the judge provided by player p is denoted j_p, the set of judges by $\mathcal{J}$, and the set of players by $\mathcal{P}$. There are four kinds of constraints in the referee assignment.

1. A judge judges exactly once a week;
2. Each group is composed of three judges;
3. Judges can grade the same player at most twice.
4. No judge can never grade his own player.

The first and second types of constraints are implicit in the algorithms presented here: they are satisfied by the initial assignments and preserved by local moves. The third and fourth sets of constraints are represented explicitly. The model contains a constraint $m[a, b]$ for every pair (a, b) of judge-player: Constraint $m[a, b]$ holds for an assignment σ if judge a does not grade player b more than twice (if judge a is not the judge furnished by player b). More precisely, if $\#_\sigma(a, b)$ denotes the number of times judge a judges player b in σ, i.e.,

$$\#\{(w, g) \mid \exists p, p' : \ \sigma(y[w, g, p]) = a \ \& \ x[w, g, p'] = b\},$$

constraint $m[a, b]$ holds if

$$\#_\sigma(a, b) \leq \begin{cases} 2 \text{ if } \ b \neq j_a; \\ 0 \text{ otherwise.} \end{cases}$$

instances	3-5-7	3-5-8	3-5-9	4-5-10	4-5-11	4-5-12	5-5-13	5-5-14	5-5-15
Nb. Add. judges	0 (50)	2 (20)	3 (80)	0 (10)	1 (90)	3	0	1 (50)	3
Failures (%)	50	20	80	10	90	0	0	50	0
Iterations	2506261.5	16559.5	5000000	76517	5000000	75173	59219	4674736.5	297947
Med. time	300.48	1.52	>1200	15.08	>1200	19.84	20.97	3458.23	128.65
Avg. time	0.48	2.39	35.86	26.88	189.48	61.98	52.68	543.39	145.8
Med. time (+1)	0.03	-	0.86	-	4.59	-	-	5.3	-

Table 3. Experimental Results on the Referee Assignment Problem.

The violations $v_\sigma(m[a,b])$ of a constraint $m[a,b]$ is the number of times judge a judges player b in schedule σ beyond their allowed meeting, i.e.,

$$v_\sigma(m[a,b]) = \max(0, \#_\sigma(a,b) - (\text{if } b \neq j_a \text{ then } 2 \text{ else } 0)).$$

As a consequence, the judge assignment problem can be modeled as the problem of finding a schedule σ minimizing the total number of violations $f(\sigma)$ where

$$f(\sigma) = \sum_{a \in \mathcal{J}, b \in \mathcal{P}} v_\sigma(m[a,b]).$$

Experimental Results Table 3 depicts the experimental results on a variety of instances related to the real-life problem. It reports the minimal number of additional judges found by SGLS for these instances, the percentage of failures of the algorithm with 5,000,000 iterations, the median number of iterations, the average time for the successful runs, and, for those instances where the failure is above 50%, the median time when an additional judge is included. First observe that SGLS solves the real-life problem with 3 additional judges in about 35 seconds in 20% of the case. For the other instances, SGLS finds solutions with no additional judge, one additional judge, or at most 3 additional judges. These instances are typically more challenging for SGLS than the earlier tournament problems, but the running times are completely acceptable. Note that the running times decrease significantly when an additional judge is included.

5 Related Work

There is a considerable body of work on scheduling social golfers in the constraint programming community. References [2],[7] describe state-of-the art results using constraint programming and are excellent starting points for more references. See also [9] for interesting theoretical and experimental results on the social golfer problem, as well as the description of SBDD, a general scheme for symmetry breaking. Reference [1] describes a tabu-search algorithm for scheduling social golfers, where the neighborhood consists of swapping the value of a single variable and where all constraints are explicit. The results are very far in quality and performance from those reported here. The neighborhood used in this paper, which implicitly maintain the group and week structures,

and the randomized tabu-list strategy are fundamental in scheduling hard instances. The algorithm for the social golfer was first presented in [3], where a special-purpose heuristic was proposed and detailed experimental results are given. The debating tournament and judge assignment problems were given to us by W. Harvey [5]. On the debating tournament, only a solution with $k = 4$ was known and the approach presented here closed the problem by providing a balanced solution with $k = 3$. No solution was known on the judge assignment problem.

6 Conclusion

This work implemented the local search algorithm SGLS for tournament scheduling and applied it to the debating tournament and judge assignment problems. The algorithm is conceptually simple: "natural" modelings of social tournaments and no use of symmetry breaking techniques. Yet the algorithm appears to be particularly effective and robust and schedules real-life applications that were open so far despite the considerable research devoted lately to this field. On debating tournaments, SGLS solves an open, real-life, instance and exhibits excellent performance over a wide variety of instances. On judge assignments, SGLS finds a high-quality solution to a real-life instance with few additional judges and is very effective on other instances as well.

7 Acknowledgements

This research was developed during a stay in Brown University, Providence, RI (USA), under the supervision of Pascal Van Hentenryck, and thus, it is a joint work with him.

Special thanks also to Warwick Harvey for many interesting discussions and for submitting these problems to us.

References

1. Ågren, M. 2003. Solving the Social Golfer Using Local Search. peg.it.uu.se/ saps02/MagnusAgren/.
2. Barnier, N., and Brisset, P. 2005. Solving kirkman's schoolgirl problem in a few seconds. *Constraints*. To appear in the Special Issue on CP'02.
3. Dot´u, I., and Van Hentenryck, P. 2004. Scheduling Social Golfers Locally. To appear in *CP-AI-OR'05*
4. Fahle, Torsten, S.; Stefan; and Sellmann, M. 2001. Symmetry breaking. In *CP-01*, 93–107.
5. Harvey, W. 2004. Personal communication.
6. Prestwich, S. 2002. Randomized backtracing for linear pseudo-boolean constraint problems. *CP-AI-OR'02*, 7–19.
7. Puget, J. 2005. Symmetry breaking revisited. *Constraints*. To appear in the Special Issue on CP'02.
8. Sellmann, M., and Harvey, W. 2002. Heuristic Constraint Propagation. *CP-AI-OR'02*.
9. Sellmann, M. 2003. *Reduction Techniques in Constraint Programming and Combinatorial Optimization*. Ph.D. Dissertation, University of Paderborn.
10. Smith, B. 2001. Reducing Symmetry in a Combinatorial Design Problem. *CP-AI-OR'2001*.