University of Reading

Scheduling reach mahjong tournaments using pseudoboolean constraints

Conference or Workshop Item

Accepted Version

Lester, M. M. ORCID: https://orcid.org/0000-0002-2323-1771 (2021) Scheduling reach mahjong tournaments using pseudoboolean constraints. In: Theory and Applications of Satisfiability Testing – SAT 2021, 5-9 July 2021, Barcelona, Spain. doi: 10.1007/978-3-030-80223-3 Available at https://centaur.reading.ac.uk/98698/

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

To link to this article DOI: http://dx.doi.org/10.1007/978-3-030-80223-3

Publisher: Springer

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

CentAUR

University of Reading

Central Archive at the University of Reading

Reading's research outputs online

Scheduling Reach Mahjong Tournaments using Pseudoboolean Constraints

Martin Mariusz Lester[0000−0002−2323−1771]

University of Reading, Reading, United Kingdom
m.lester@reading.ac.uk

Abstract. Reach mahjong is a gambling game for 4 players, most popular in Japan, but played internationally, including in amateur tournaments across Europe. We report on our experience of generating tournament schedules for tournaments hosted in the United Kingdom using pseudoboolean solvers. The problem is essentially an extension of the well-studied Social Golfer Problem (SGP) in operations research. However, in our setting, there are further constraints, such as the positions of players within a group, and the structure of the tournament graph, which are ignored in the usual formulation of the SGP. We tackle the problem primarily using the SAT/pseudoboolean solver clasp, but sometimes augmented with an existing local search-based solver for the SGP.

Keywords: social golfer problem · mahjong · tournament scheduling · pseudoboolean constraints

Reach mahjong (or *riichi* mahjong) is a gambling game for 4 players. A game (or *hanchan*) is played over several rounds. In each round, players seated at a table sequentially draw and discard tiles in an attempt to form a winning hand of 14 tiles. At the end of the round, the losing players pay a number of points to the winner, according to the value of his hand. The player with the most points at the end of the game is the winner.

Reach mahjong is most popular in Japan, although it is played throughout the world. In Europe, a few hundred amateur players compete in tournaments arranged throughout the year and around the continent. Tournaments are typically organised locally, but run following rules published by the European Mahjong Association (EMA) [1], which has approved and ranked tournaments since 2008. This raises the question of how best to schedule games in a tournament.

We report on our experience of using the pseudoboolean (PB) solver *clasp* [8] to generate tournament schedules (such as Table 1) for tournaments run in the United Kingdom since 2013. This includes generating a schedule for 128 players over 10 sessions for the 2016 European Riichi Mahjong Championship (ERMC), which satisfied a complex combination of constraints. Our software CoMaToSe (Constraint Mahjong Tournament Scheduler) and benchmarks are online [13].

In Section 1 we describe some details of the tournament scheduling problem and the constraints that they lead to. Then, in Section 2, we describe how we encode those constraints. The scheduling problem is essentially an extension

of the Social Golfer Problem (SGP), which is amenable to solution using SAT solvers [9, 12], but we found that for larger tournaments, and with our extended set of constraints, a PB formulation was more tractable. To our knowledge, there is no previous published work considering mahjong tournament scheduling; our encoding of the constraints extending the SGP is original. We evaluate our approach in Section 3. Finally, in Section 4 we discuss related work on the SGP, and more generally on tournament scheduling for games with more than 2 players, before concluding.

1 Problem Description

Size, length and format. A number of practical constraints and conventions have arisen over time that essentially fix the format of a tournament. Tournaments typically take place at the weekend, so usually last 1–2 days. Players may travel to a tournament and expect to play throughout; tournament rules specify that all players should play the same number of games and, after adding up each player's score in every game, the player with the highest score should be the winner. Games are played to a time limit in a session of 90 minutes. Allowing time for breaks and so on, it is usual to have 3–5 sessions in a day, for a total of 4–10 sessions in a tournament. A 2-day tournament might typically attract around 48 players. Every 3 years, a European Championship (ERMC) is held, which lasts longer and attracts more players. The main phase of the 2016 championship in the UK had 128 players over 10 sessions; the 2019 championship in the Netherlands had 140 players over 12 sessions. A tournament venue usually has enough tables and equipment to allow all participants to play simultaneously, although this may be variable in quality.

Wind allocation. During each round, each player is allocated a different compass point (East, South, West or North) as his *wind*. The East player is the *dealer*. When he wins, he gets 50% more points and stays as dealer for the next round. If he does not win, the wind allocations rotate. He becomes North for the next round, the South player becomes East, and so on. In a full game, each player gets to be East twice; being North initially (and hence East last) may be an advantage because it confers some control over when the game ends. However, if a game is curtailed because of a tournament time limit, being North or West initially puts one at a disadvantage, as one often misses a second turn as dealer.

Table 1. A tournament schedule for 24 players and 5 sessions (SGP 6-4-5) with $d = 10$.

Table	1				2				3				4				5				6			
Session	E	S	W	N	E	S	W	N	E	S	W	N	E	S	W	N	E	S	W	N	E	S	W	N
1	1	3	2	4	7	8	5	6	11	12	10	9	14	13	15	16	17	20	18	19	24	22	23	21
2	9	23	7	17	3	19	10	15	16	6	24	2	8	18	21	12	22	5	1	14	20	11	4	13
3	11	18	14	24	21	1	13	17	20	7	3	22	10	4	6	23	2	15	9	8	12	16	19	5
4	15	21	6	20	4	22	16	9	19	23	8	14	2	17	11	5	3	24	12	13	18	10	1	7
5	13	8	22	10	23	2	20	12	5	15	4	18	19	9	24	1	7	16	21	11	6	14	17	3

The core problem is thus how best to schedule 4–12 sessions of a 4-player game for 16–140 players. Initial wind allocations for a table can be included in the schedule, or they can be drawn at random by the players before the game. Players might feel aggrieved if they have a disadvantageous wind allocation, even if it was produced at random. Balanced allocation of winds in the schedule avoids this and reduces setup time at the start of the game.

1.1 Constraints

The following have been suggested as desirable properties in a schedule:

Socialisation: Players want to play as many different opponents as possible.
Wind balance: In order to share the potential penalty of not getting a second turn as dealer across all players, each player should be allocated each starting wind position roughly an equal number of times. Obviously, this is not possible when the number of sessions is not a multiple of 4 and even then, it may not always be possible.
Table movement: To reduce the chances of cheating, and to share the inconvenience of playing on a table with low-quality equipment, a player should not play on the same table twice.

Of these, most players consider socialisation most important. With just this requirement, tournament scheduling is an instance of the well-studied SGP. With the relatively low ratio of players to sessions in many tournaments, satisfying just this requirement leaves relatively little scope for changing who plays who in each round. However, in larger tournaments, we may also wish to consider what properties are desirable in the tournament graph of "who plays who". Intuitively, we can view a tournament as a process by which points flow along edges from losing players to winning players. We then expect that the final scores of players are indicative of a linear ordering of their skill, but this depends on there being adequate potential for points to flow between any two players. This leads to the following desirable properties of the tournament graph:

Graph connectedness: The tournament graph should be connected.
Graph diameter: The diameter of the tournament graph (greatest distance between any pair of players) of the tournament should be as low as possible. In our setting, this usually means 2. This is a stronger version of the requirement that the graph be connected, with the same motivation. If two players cannot face each other directly, points can still flow between them via other players, but we would like the route to be as short as possible.
Multiple short paths: If two players in the tournament graph are not adjacent, there should be multiple paths between them, ideally of length 2. This increases the potential for indirect flow of points between them.

For a tournament graph with diameter 2, we will refer to the minimum number of paths of length 2 between any two non-adjacent players in the tournament graph as d. Necessarily, $d \geq 1$. For the schedule in Table 1, we used graph constraints to enforce $d = 10$; prior to this, the schedule had $d = 9$.

2 Problem Encoding

As the most important constraint is socialisation, we start with a PB encoding for the SGP, then add our other constraints in a monolithic formulation. By convention, we refer to an SGP instance as *g-p-w* with:

- g — the number of *groups* playing simultaneously
- p — the number of *players* in a group (always 4 for mahjong)
- w — the number of sessions (*weeks* in the SGP)

We chose our solver by evaluating participants in the SAT and PB [4] competitions of 2012, which were the most recent competitions at the time. Of these, we found that *clasp* [8] was most effective, particularly when run with the *crafty* preset for combinatorially hard problems.

Initially, we had focused on the use of SAT, as Triska had developed an effective SAT encoding of the SGP [18]. However, his socialisation constraint is $\mathcal{O}(g^4p^2w^2)$. For large numbers of players, we found that just generating the constraints was too slow, so starting with the 2016 European Championship (SGP 32-4-10), we adopted a PB formulation.

We had assumed that, while a PB might not find an optimal solution, if it ceased to make progress, it would at least have found a locally optimal solution. However, a tournament organiser told us he had been able to improve wind balance in a generated schedule by shuffling wind allocations of two tables. We found that, using an encoding that considered only wind balance, we could fine-tune the schedule generated from the monolithic encoding and automate this.

Although the SAT/PB method for solving SGP instances is competitive, the best automated method currently known is Triska's heuristic-guided local search algorithm [17]; an implementation by Rezaei is available online [15]. Therefore, for tournament schedules that correspond to hard instances of the SGP, we can import a solution to the SGP instance and just tune the wind balance. While fixing group allocations in this way may remove the best solutions from the space considered by the solver, in practice it allows us to find better solutions than using a constraint solver in isolation. In all cases we considered, we were able to find an optimal wind allocation this way.

2.1 Monolithic Constraint Encoding

We now present our monolithic PB constraint encoding of the problem. We set $n = g.p$ as the number of players. The constraints range over the following Boolean variables, where $h, i, j \in [1, n]$ with $j > i$, $h \neq i$ and $h \neq j$, $k \in [1, g]$, $l \in [1, w]$ and $s \in [1, p]$:

- $P_{i,k,l}$ — true just if i plays in group k in session l
- $S_{i,k,l,s}$ — true just if i plays in group k in session l in seat position s
- $M_{i,j,l}$ — true if i and j meet in session l
- $C_{i,j}$ — true only if i and j meet (compete) in any session
- $D_{i,j,h}$ — true only if i and j both meet h (compete *indirectly*)

We encode East as position 1, South as 2 and so on. Constraint sets are as follows; quantification ($\forall$, $\sum$) of indices is always implicitly over the ranges above.

Each group must have exactly p players:

$$\forall k, l. \sum_i P_{i,k,l} = p \tag{1}$$

Each player must play in exactly one group in each session:

$$\forall i, l. \sum_k P_{i,k,l} = 1 \tag{2}$$

Optionally, to break symmetries, order players sequentially in the first session:

$$\forall i. P_{i,\lceil i/g \rceil,1} = 1 \tag{3}$$

If i and j play in the same group in the same session, then they must meet in that session:

$$\forall i, j, k, l. -P_{i,k,l} + -P_{j,k,l} + M_{i,j,l} \geq -1 \tag{4}$$

and they must meet at most once over all sessions:

$$\forall i, j. \sum_l -M_{i,j,l} \geq -1 \tag{5}$$

i and j competed only if they played in the same group in any session:

$$\forall i, j. -C_{i,j} + \sum_{k,l} P_{i,k,l} P_{j,k,l} \geq 0 \tag{6}$$

i and j competed indirectly via h only if they both competed with h:

$$\forall i, j, h. C_{\min(i,h),\max(i,h)} + C_{\min(h,j),\max(h,j)} + -2D_{i,j,h} \geq 0 \tag{7}$$

i and j must compete directly, or compete indirectly d times (d configurable):

$$\forall i, j. d \cdot C_{i,j} + \sum_h D_{i,j,h} \geq d \tag{8}$$

Each player must play in each group at most once over all sessions:

$$\forall i, k. \sum_l -P_{i,k,l} \geq -1 \tag{9}$$

If i sits in a position in a group, he must play in that group:

$$\forall i, k, l, s. -S_{i,k,l,s} + P_{i,k,l} \geq 0 \tag{10}$$

If i plays in a group, he must sit in one of its positions:

$$\forall i, k, l. -P_{i,k,l} + \sum_s S_{i,k,l,s} \geq 0 \tag{11}$$

Exactly one player must sit in every seat:

$$\forall k, l, s. \sum_i S_{i,k,l,s} = 1 \tag{12}$$

Each player must play in each position (roughly) the same number of times:

$$\forall i, s. \sum_{k,l} S_{i,k,l,s} \geq \lfloor w/p \rfloor \qquad \forall i, s. \sum_{k,l} -S_{i,k,l,s} \geq -\lceil w/p \rceil \tag{13}$$

Constraints 1–5 follow Walser [20]; the rest are original. The constraint sets are largely orthogonal: any of 4–5 (socialisation), 6–8 (enforcing d), 9 (table movement) and 10–13 (wind balance) can be removed independently. Note constraints 6 are non-linear. Concerning size: 4–5 is $\mathcal{O}(g^2p^2w)$ ($\mathcal{O}(g^2w)$ smaller than Triska's SAT encoding); 6–8 is $\mathcal{O}(g^3p^3)$; 10–13 is $\mathcal{O}(g^2p^2w)$.

In practice, it may not always be possible to satisfy all constraints simultaneously, whether because there is no solution, or because the solver cannot find one. In these cases, constraint sets 5, 8, 9 or 13 can be made soft, turning the problem into a Weighted Boolean Optimisation (WBO) instance.

Apart from the obvious symmetry breaking of fully specifying session 1, most existing symmetry breaking techniques for the SGP violate the extra constraints in our problem. For example, putting players 1-4 on tables 1-4 in later rounds, or requiring that the tables are ordered by lowest numbered player on the table, violates table movement. Formulations of the pure SGP that encode a table as an ordered list usually benefit from breaking symmetry in the ordering of players. In the PB formulation, native cardinality constraints make it easy to encode a table as an unordered set, so there is no symmetry to break. Of course, when one adds wind allocation, ordering of players at a table is no longer a symmetry.

2.2 Wind Balancing Constraint Encoding

Our constraint encoding for fine-tuning wind allocations uses variables $W_{i,l,s}$, which are true just if player i is in position s in session l. The constraints depend on a fixed allocation of players to groups, which we refer to using values of P variables from the monolithic encoding; tuning can only change a player's seat at a table. Our encoding is as follows. Each player must have a seat:

$$\forall i, l. \sum_{s} W_{i,l,s} = 1 \tag{14}$$

Exactly one player in a group can take each seat:

$$\forall k, l, s. \sum_{\{i|P_{i,k,l}\}} W_{i,l,s} = 1 \tag{15}$$

Each player must play in each position (roughly) the same number of times:

$$\forall i, s. \sum_{l} W_{i,l,s} \geq \lfloor w/p \rfloor \qquad \forall i, s. \sum_{l} -W_{i,l,s} \geq -\lceil w/p \rceil \tag{16}$$

3 Evaluation

We have used our encoding to generate schedules for the 2016 ERMC and several smaller tournaments in the UK. Timings were generated on a machine running Debian Linux 10 with a 3.4 GHz Intel Core i5-7500 CPU and 64 GB of RAM. We used *clasp* 3.3.4 with *crafty* preset and Rezaei's local search SGP solver [15].

For the 2016 ERMC (32-4-10), we used our monolithic encoding, incrementally turning on constraints to obtain the best schedule possible. Enforcing just socialisation took 18s. We turned on table movement and wind balance, tightening the wind constraints (13) to give each player 2 turns in each seat plus 1 turn as East or South and 1 turn as West or North; solving this took 2m10s. The schedule's tournament graph already had diameter 2, but $d = 1$. Adding the constraint $d = 2$, it took 14m to solve. Changing to $d = 3$, *clasp* found no solution in 1h. So finally, keeping the hard constraint $d = 2$ and adding a soft constraint $d = 3$, with a timeout of 1h, we generated a schedule violating only 122 of $\binom{128}{2} = 8128$ soft constraints. Overall, the instance had 1.3M variables and 3.9M constraints. Appendix A shows benchmarks for similar instances.

For comparison, solving the SGP instance with local search and balancing the winds using our constraint formulation yielded a solution in less than 1s. The tournament graph had diameter 2, but with $d = 1$, and there is no easy way to tune the graph while maintaining socialisation.

For the smaller tournaments, the tournament graph was necessarily low diameter, so we did not enforce it with constraints. 1-day tournaments usually had 5 sessions and ranged from 24 to 52 players (6-4-5 to 13-4-5). 2-day tournaments usually had 8 sessions and ranged from 32 to 68 players (8-4-8 to 17-4-8). We benchmarked our monolithic encoding on these intervals, setting a solver time limit of 10m and making wind balance a soft constraint. For instances in the 1-day interval, it took less than 0.5s to solve constraints for a schedule with maximal socialisation, table movement and wind balance, except for 6-4-5, which took 3.6s. For the 2-day interval (see Table 2), the 8-4-8 instance is significant as it is the original formulation of the SGP, and remains out of reach for SAT- and PB-based methods, including ours, so we imported a solution to balance. For 9-4-8 and 10-4-8, *clasp* solved the constraints only with wind balance turned off. For the rest of the interval, *clasp* found solutions with 2–21 wind constraint violations. In all cases, whether using schedules generated by our monolithic encoding, or importing schedules generated by local search, we were able to tune wind balance perfectly, satisfying all constraints, usually in under 2s.

To confirm that *clasp* was still an appropriate choice of solver, we compared up-to-date versions of entrants in the relevant track (WBO SOFT-SMALLINT-NLC) of the most recent PB Competition (2016), as well as experimental versions of *SAT4J* using the cutting planes and rounding SAT techniques. Table 3 shows

Table 2. Benchmarks applying monolithic encoding to 2-day tournaments (g-4-8).

Groups	8	9	10	11	12	13	14	15	16	17
Variables	14k	18k	22k	27k	32k	37k	44k	50k	57k	64k
Constraints	43k	60k	81k	105k	134k	168k	207k	252k	304k	361k
Socialisation only time (s)	-	53	0.52	0.46	0.56	0.77	0.94	1.2	1.5	1.7
Constraints violated after 10m	-	-	-	2	3	3	3	13	11	21
Total wind constraints	-	-	-	176	192	208	224	240	256	272
Wind balance tuning time (s)	0.045	1.2	0.094	0.15	0.17	0.030	1.5	0.13	15	9.5

the comparison. Although the standard *SAT4J* solver is competitive, *clasp* is still best. Some new PB solvers have participated in the more active MaxSAT Evaluation competition, but none supports WBO with non-linear clauses.

Summary: For large instances, where the tournament graph structure was of concern, our monolithic constraint encoding allowed the graph to be optimised at the same time as allocating wind positions. For hard SGP instances, we necessarily had to import an SGP solution, but our wind encoding successfully tuned this. In other cases, there was little difference between the quality of schedules generated: using our monolithic encoding, then tuning wind allocations if necessary; and using a local search SGP solver, followed by tuning wind allocation. However, the latter was considerably faster.

4 Related Work and Conclusions

The SGP was posted on the Usenet group sci.op-research in 1998. The original SGP, to find the highest w for which 8-4-w is solvable, is problem 10 in CSPLib [10]. Optimal solutions of the SGP are entry A107431 in OEIS [2]. Walser suggested a PB encoding [20]. Later, Gent and Lynce proposed a SAT encoding [9]. Much work on solving SGP instances focuses on breaking symmetries [7, 11, 6, 3]. Triska studied the problem extensively [18, 17, 16].

Recently, Lardeux and others revisited the SAT encoding, exploring efficient, correct translation of set constraints [12]; they seem unaware of PB problems/solvers. Liu and others investigated solving SGP instances in parallel [14].

We found no previous research specifically on scheduling a mahjong tournament. Bridge and whist are 4-player games, but played by 2 co-operating pairs, not 4 competing individuals. Individual bridge tournaments [19] were played in the past, but are currently not popular. Whist games are shorter than mahjong games, and partnership is still significant, which leads to different goals [5].

We have shown how to generate good tournament schedules for reach mahjong tournaments run according to the conventions of the European Mahjong Association. Since 2013, our approach has been used to generate schedules for several tournaments hosted in the UK, including the 2016 ERMC. Our experience reaffirms the message that SAT/PB solvers are an effective and convenient but imperfect tool for solving complex problems that arise in real life. We think it is likely that a custom local search algorithm that considered all our constraints simultaneously would outperform our approach. However, this would have been far less convenient than applying an existing solver.

Table 3. Comparison of NLC WBO solvers. Constraints violated after 10m (g-4-8).

Groups	8	9	10	11	12	13	14	15	16	17
clasp	-	-	-	2	3	3	3	13	11	21
SAT4J	-	-	-	36	9	8	10	16	20	28
NaPS	-	-	-	-	38	30	46	67	49	73
ToySat	-	-	-	-	-	-	-	-	-	-

Groups	8	9	10	11	12	13	14	15	16	17
SAT4J-cutting	-	-	-	-	-	-	-	-	65	69
SAT4J-rounding	-	-	-	-	-	-	-	100	-	86
SAT4J-partial	-	-	-	-	-	-	-	-	-	76

A Benchmarks for Large Instances

Table 4. Benchmarks applying monolithic encoding to large tournaments (g-4-10).

Groups		28	29	30	31	32	33	34	35	36
Soc. + wind time (s)		2.7	3.1	4.2	3.3	131	5.3	102	76	241
Soc. + wind + $d = 2$ time (s)		46	46	38	100	847	664	1189	-	-
With $d = 3$ soft:	Variables	0.9M	1.0M	1.1M	1.2M	1.3M	1.4M	1.5M	1.7M	1.8M
	Constraints	2.6M	2.9M	3.2M	3.5M	3.8M	4.2M	4.6M	5.0M	5.5M
Constraints violated after 1h		0	62	61	1	122	196	-	327	-

References

1. European Mahjong Association, http://mahjong-europe.org/
2. The on-line encyclopedia of integer sequences. https://oeis.org/A000108, sequence A000108
3. Azevedo, F.: An attempt to dynamically break symmetries in the social golfers problem. In: Azevedo, F., Barahona, P., Fages, F., Rossi, F. (eds.) Recent Advances in Constraints, 11th Annual ERCIM International Workshop on Constraint Solving and Contraint Logic Programming, CSCLP 2006, Caparica, Portugal, June 26-28, 2006, Revised Selected and Invited Papers. Lecture Notes in Computer Science, vol. 4651, pp. 33–47. Springer (2006). https://doi.org/10.1007/978-3-540-73817-6_2, https://doi.org/10.1007/978-3-540-73817-6_2
4. Balint, A., Belov, A., Järvisalo, M., Sinz, C.: Overview and analysis of the SAT challenge 2012 solver competition. Artif. Intell. **223**, 120–155 (2015). https://doi.org/10.1016/j.artint.2015.01.002, https://doi.org/10.1016/j.artint.2015.01.002
5. Berman, D.R., McLaurin, S.C., Smith, D.D.: Ranking whist players. Discret. Math. **283**(1-3), 15–28 (2004). https://doi.org/10.1016/j.disc.2004.01.005, https://doi.org/10.1016/j.disc.2004.01.005
6. Cotta, C., Dotú, I., Fernández, A.J., Hentenryck, P.V.: Scheduling social golfers with memetic evolutionary programming. In: Almeida, F., Aguilera, M.J.B., Blum, C., Moreno-Vega, J.M., Pérez, M.P., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics, Third International Workshop, HM 2006, Gran Canaria, Spain, October 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4030, pp. 150–161. Springer (2006). https://doi.org/10.1007/11890584_12, https://doi.org/10.1007/11890584_12
7. Dotú, I., Hentenryck, P.V.: Scheduling social golfers locally. In: Barták, R., Milano, M. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Second International Conference, CPAIOR 2005, Prague, Czech Republic, May 30 - June 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3524, pp. 155–167. Springer (2005). https://doi.org/10.1007/11493853_13, https://doi.org/10.1007/11493853_13
8. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: *clasp* : A conflict-driven answer set solver. In: Baral, C., Brewka, G., Schlipf, J.S. (eds.) Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007,

Tempe, AZ, USA, May 15-17, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4483, pp. 260–265. Springer (2007). https://doi.org/10.1007/978-3-540-72200-7_23, https://doi.org/10.1007/978-3-540-72200-7_23
9. Gent, I.P., Lynce, I.: A sat encoding for the social golfer problem. In: In IJCAI'05 workshop on Modelling and Solving Problems with Constraints (2005), https://www.inesc-id.pt/ficheiros/publicacoes/2516.pdf
10. Harvey, W.: CSPLib problem 010: Social golfers problem. http://www.csplib.org/Problems/prob010
11. Harvey, W., Winterer, T.J.: Solving the MOLR and social golfers problems. In: van Beek, P. (ed.) Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3709, pp. 286–300. Springer (2005). https://doi.org/10.1007/11564751_23, https://doi.org/10.1007/11564751_23
12. Lardeux, F., Monfroy, E., Crawford, B., Soto, R.: Set constraint model and automated encoding into SAT: application to the social golfer problem. Ann. Oper. Res. **235**(1), 423–452 (2015). https://doi.org/10.1007/s10479-015-1914-5, https://doi.org/10.1007/s10479-015-1914-5
13. Lester, M.M.: CoMaToSe: Constraint Mahjong Tournament Scheduler (May 2021). https://doi.org/10.5281/zenodo.4764650, https://doi.org/10.5281/zenodo.4764650
14. Liu, K., Löffler, S., Hofstedt, P.: Social golfer problem revisited. In: van den Herik, H.J., Rocha, A.P., Steels, L. (eds.) Agents and Artificial Intelligence - 11th International Conference, ICAART 2019, Prague, Czech Republic, February 19-21, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11978, pp. 72–99. Springer (2019). https://doi.org/10.1007/978-3-030-37494-5_5, https://doi.org/10.1007/978-3-030-37494-5_5
15. Rezaei, A.: Golfer: A toolkit for solving social golfer problem (2015), https://github.com/arezae4/golfer
16. Triska, M.: Solution methods for the social golfer problem (2008), https://www.metalevel.at/mst.pdf, Master's thesis
17. Triska, M., Musliu, N.: An effective greedy heuristic for the social golfer problem. Ann. Oper. Res. **194**(1), 413–425 (2012). https://doi.org/10.1007/s10479-011-0866-7, https://doi.org/10.1007/s10479-011-0866-7
18. Triska, M., Musliu, N.: An improved SAT formulation for the social golfer problem. Ann. Oper. Res. **194**(1), 427–438 (2012). https://doi.org/10.1007/s10479-010-0702-5, https://doi.org/10.1007/s10479-010-0702-5
19. English Bridge Union: Individual competitions, https://www.ebu.co.uk/documents/cmh/Individuals.pdf
20. Walser, J.P.: AMPL model of 'maximum socializing on the golf course'. https://www.csplib.org/Problems/prob010/models/AMPLmodel.txt.html (1998)