Ann Oper Res (2012) 194:427–438
DOI 10.1007/s10479-010-0702-5

An improved SAT formulation for the social golfer problem

Markus Triska · Nysret Musliu

Published online: 5 February 2010

Abstract The Social Golfer Problem (SGP) is a sports scheduling problem that exhibits a lot of symmetry and has recently attracted significant attention. In this paper, we first revisit an existing SAT encoding for the SGP and correct some of its clauses. We then propose a change in the encoding that significantly reduces the number of variables for all instances. We achieve considerable performance improvements when solving many SGP instances with common SAT solvers using local search and complete backtracking. This makes SAT formulations a more promising approach for solving the SGP than previously.

Keywords Sports scheduling · Combinatorial optimization · Design theory · Finite geometry

1 Introduction

The *Social Golfer Problem* (SGP) is a sports scheduling problem derived from a question that was posted to `sci.op-research` in May 1998. It is problem number 10 in CSPLib, a benchmark library for constraints (Gent and Walsh 1999):

> 32 golfers play golf once a week, and always in groups of 4. For how many weeks can they play such that no two players play together more than once in the same group?

The problem is readily generalized to the following decision problem: Is it possible to schedule $n = g \times p$ golfers in g groups of p players for w weeks such that no two golfers

The research herein is partially conducted within the competence network Softnet Austria (www.soft-net.at) and funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsförderungsgesellschaft mbH. (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

M. Triska (✉) · N. Musliu
Database and Artificial Intelligence Group, Vienna University of Technology, Vienna, Austria
e-mail: triska@dbai.tuwien.ac.at

N. Musliu
e-mail: musliu@dbai.tuwien.ac.at

play in the same group more than once? An *instance* of the SGP is then denoted by a triple $g-p-w$ of natural numbers. In practice, one is interested in the corresponding optimization problem that asks for the maximum number w^* of weeks that admits a solution for given g and p, since such a solution also implies solutions for all instances $g-p-w$ with $w \leq w^*$.

The SGP exhibits many symmetries: Weeks, groups within weeks, and players within groups, can all be ordered arbitrarily. In addition, players can be assigned arbitrary names. Due to its highly constrained and symmetric nature, the SGP has attracted much attention from the constraint programming community and has led to the development of powerful but complex dynamic symmetry breaking schemes (Petrie and Smith 2004; Barnier and Brisset 2005). In addition to being an interesting and hard benchmark problem, the SGP and closely related problems have many important practical applications, such as in encoding, encryption and covering problems. Many instances of the SGP are still open, and the continuing progress of SAT solvers makes it attractive to try a SAT formulation of the SGP. Gent and Lynce have already proposed a SAT encoding (Gent and Lynce 2005). We first revisit their formulation. Then, we change the formulation to significantly reduce the number of variables for all instances. We show that this reduces computation time with common SAT solvers considerably.

2 Related work

Research on a problem that is very closely related to the SGP actually dates back to Euler, who considered an instance of the SGP in a different context: Euler asked whether two orthogonal Latin squares of order 6 exist (Colbourn and Dinitz 1996), which has become known as "Euler's Officer Problem". In terms of the SGP, this corresponds to solving the $6-6-4$ instance, which is now known to be impossible. As another special case of the SGP, the $5-3-7$ instance also has a long history and is known as Kirkman's schoolgirl problem (Barnier and Brisset 2005).

The computational complexity of the SGP is currently unknown. Some instances are easily solved using construction methods from *design theory*, a branch of discrete mathematics, but such methods are typically restricted to certain families of instances. From a result derived by Colbourn (1984), one can show that the *completion problem* of the SGP, i.e., deciding whether a partially filled schedule can be completed to conflict-free one, is NP-complete.

In general, the task of finding w mutually orthogonal Latin squares (MOLS) of order q is equivalent to solving the SGP instance $q-q-(w+2)$ (Anderson 1997). MOLS play an important rôle in the design of statistical experiments, coding theory and cryptography, and several (in)existence results and construction methods for specific instances are known from design theory. Harvey and Winterer have compared and successfully applied several of these interesting techniques to the SGP (Harvey and Winterer 2005).

Metaheuristic approaches towards the SGP include local search with tabu-lists (Dotú and Hentenryck 2005), and an evolutionary approach (Cotta et al. 2006).

3 The SAT formulation by Gent and Lynce

Consider the general $g-p-w$ instance of the SGP, with $x = g \times p$ the number of golfers. For their SAT formulation (Gent and Lynce 2005), Gent and Lynce introduce variables G_{ijkl} $(1 \leq$

$i \le x$, $1 \le j \le p$, $1 \le k \le g$ and $1 \le l \le w$) denoting whether player i plays in position j in group k and week l. The constraints are then imposed by a set of clauses ensuring that:

- Each player plays exactly once per week, i.e.:
  - Each player plays *at least* once per week
  - Each player plays *at most* once per week
- Each group consists of exactly p players, i.e.:
  - *At least* one player is the jth golfer ($1 \le j \le p$)
  - *At most* one player is the jth golfer ($1 \le j \le p$)
- No two players play in the same group more than once

The following clauses ensure that each player plays *at least* once in each week:

$$\bigwedge_{i=1}^{x}\bigwedge_{l=1}^{w}\bigvee_{j=1}^{p}\bigvee_{k=1}^{g}\mathrm{G}_{ijkl}. \tag{1}$$

To enforce that each player plays *at most* once each week, it is first ensured that each player plays at most once *per group* in each week:

$$\bigwedge_{i=1}^{x}\bigwedge_{l=1}^{w}\bigwedge_{j=1}^{p}\bigwedge_{k=1}^{g}\bigwedge_{m=j+1}^{p}\neg\mathrm{G}_{ijkl}\vee\neg\mathrm{G}_{imkl}. \tag{2}$$

A second set of clauses is supposed to guarantee that no player plays in more than one group in any week:

$$\bigwedge_{i=1}^{x}\bigwedge_{l=1}^{w}\bigwedge_{j=1}^{p}\bigwedge_{k=1}^{g}\bigwedge_{m=k+1}^{g}\bigwedge_{n=j+1}^{p}\neg\mathrm{G}_{ijkl}\vee\neg\mathrm{G}_{inml}. \tag{3}$$

However, clause set (3) as proposed is incomplete, and we return to it below. Taking our modification below into account, the clause set (1) ∪ (2) ∪ (3) enforces that each player plays exactly once per week.

A similar set of clauses is introduced for *groups* of golfers:

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{j=1}^{p}\bigvee_{i=1}^{x}\mathrm{G}_{ijkl}, \tag{4}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{j=1}^{p}\bigwedge_{i=1}^{x}\bigwedge_{m=i+1}^{x}\neg\mathrm{G}_{ijkl}\vee\neg\mathrm{G}_{imkl}. \tag{5}$$

The set (4) ∪ (5) is intended to yield valid groups, i.e., groups where exactly one player is in position j for each $1 \le j \le p$. We will return to this clause set below as well.

The only constraint left to encode is "socialisation", i.e., no two players can play in the same group more than once. To express this constraint, Gent and Lynce introduce a set of auxiliary variables and a so-called *ladder* matrix. The auxiliary variables G'_{ikl} ($1 \le i \le x$, $1 \le k \le g$ and $1 \le l \le w$) denote whether player i plays in group k and week l. They are related to the variables G_{ijkl} via the equivalence:

$$\mathrm{G}'_{ikl} \leftrightarrow \bigvee_{j=1}^{p}\mathrm{G}_{ijkl} \tag{6}$$

Fig. 1 Ladder matrix for the schedule of Fig. 2

	1.1	1.2	2.1	2.2	3.1	3.2
3.4	T	**T**	F	F	F	F
2.3	T	T	T	T	T	**T**
2.4	T	T	T	**T**	F	F
1.2	**T**	F	F	F	F	F
1.3	T	T	**T**	F	F	F
1.4	T	T	T	T	**T**	F

Fig. 2 Schedule corresponding to the ladder matrix of Fig. 1

	Week 1		**Week 2**		**Week 3**	
Group 1	1	2	1	3	1	4
Group 2	3	4	2	4	2	3

posted for all $1 \leq i \leq x$, $1 \leq k \leq g$ and $1 \leq l \leq w$.

The ladder matrix is a $\binom{x}{2} \times (g \times w)$ roster of propositional variables denoted by $\textsc{Ladder}_{yz}$. A complete assignment of the ladder variables is said to be *valid* iff every row is a sequence of zero or more TRUE assignments followed by only FALSE assignments. To enforce this, Gent and Lynce propose the set of clauses:

$$\bigwedge_{y=1}^{\binom{x}{2}-1} \bigwedge_{z=1}^{g\times w} \neg\textsc{Ladder}_{yz+1} \vee \textsc{Ladder}_{yz}. \tag{7}$$

We will return to this clause set below. For now, we illustrate the intention of the ladder matrix with an example given in Fig. 1. It is taken from Gent and Lynce (2005) and corresponds to the schedule shown in Fig. 2. Each row of the matrix corresponds to a pair of golfers. In each row, the column of the rightmost TRUE value, if any, denotes the group in which the respective two golfers play together. We highlight the rightmost TRUE value of each row using a bold **T**. Obviously, at most one TRUE assignment can be the rightmost one in each row. Therefore, each pair of players can occur in at most one group.

The remaining sets of clauses relate the auxiliary variables G'_{ikl} with the ladder variables. If two golfers i and m play in the same group, i.e., $G'_{ikl} \wedge G'_{mkl}$ is true for $i < m$, then, according to Gent and Lynce, $\textsc{Ladder}_{[\binom{x-i}{2}+m-i](l\times k)}$ must be TRUE and $\textsc{Ladder}_{[\binom{x-i}{2}+m-i](l\times k+1)}$ must be FALSE, and conversely. Formally, they propose the following sets of clauses:

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \neg G'_{ikl} \vee \neg G'_{mkl} \vee \textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k}, \tag{8}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \neg G'_{ikl} \vee \neg G'_{mkl} \vee \neg\textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k+1}, \tag{9}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k+1} \vee \neg\textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k} \vee \neg G'_{ikl}, \tag{10}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k+1} \vee \neg\textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k} \vee \neg G'_{mkl}. \tag{11}$$

We discuss these clauses in the next section.

4 Revisiting the SAT formulation by Gent and Lynce

In this section, we revisit the SAT formulation as proposed by Gent and Lynce and correct some of its clauses, to enforce all desired constraints and thus correctly model SGP instances as SAT instances.

Clauses (1) and (2) of the SAT formulation by Gent and Lynce are correct. Clause set (3) is incomplete: n must range from **1** to p, instead of from $j+1$ to p. This is because a player that plays in week l at position j of group k must not play in *any* other position for further groups of that week l, not just positions *greater* than j. The correct encoding is thus:

$$\bigwedge_{i=1}^{x}\bigwedge_{l=1}^{w}\bigwedge_{j=1}^{p}\bigwedge_{k=1}^{g}\bigwedge_{m=k+1}^{g}\bigwedge_{n=1}^{p}\neg \mathrm{G}_{ijkl}\vee\neg \mathrm{G}_{inml}. \tag{12}$$

Clause set (5) is slightly inaccurate: G_{imkl} must be changed to $\mathrm{G}_{\mathbf{mj}kl}$ to enforce the correct constraint, yielding:

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{j=1}^{p}\bigwedge_{i=1}^{x}\bigwedge_{m=i+1}^{x}\neg \mathrm{G}_{ijkl}\vee\neg \mathrm{G}_{mjkl}. \tag{13}$$

This is because the other order of these indices would not match the intended usage of these variables.

Clause set (7) is also slightly inaccurate: y must range from 1 to $\binom{x}{2}$. This is because the constraint must hold for *all* pairs of players, not just every one except the last one, and yields:

$$\bigwedge_{y=1}^{\binom{x}{2}}\bigwedge_{z=1}^{g\times w}\neg \textsc{Ladder}_{yz+1}\vee \textsc{Ladder}_{yz}. \tag{14}$$

It also becomes clear from this clause set that the ladder matrix is in fact a $\binom{x}{2}\times(g\times w+\mathbf{1})$ matrix.

Further, it is easy to see that in the clauses of (10), $\neg \mathrm{G}'_{ikl}$ must be replaced by its negation, G'_{ikl}. Similarly, $\neg \mathrm{G}'_{mkl}$ must be replaced by its negation in the clauses of (11), yielding:

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x}\textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k+1}\vee\neg \textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k}\vee \mathrm{G}'_{ikl}, \tag{15}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x}\textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k+1}\vee\neg \textsc{Ladder}_{\binom{x-i}{2}+m-i,l\times k}\vee \mathrm{G}'_{mkl}. \tag{16}$$

This is due to the intention that if adjacent cells of the ladder matrix are TRUE and FALSE in some row, the two players corresponding to that row *should* play together.

However, this is not all that needs to be changed to correctly model the SGP. Consider the "solution" of the SGP instance 8–4–2 shown in Fig. 3, in which conflict positions are highlighted. The configuration satisfies all constraints corrected so far, yet still contains conflicts.

The location of conflict positions in Fig. 3 can give valuable indications as to where the model went wrong. In this case, we expect the encoding of *socialisation* to be wrong, since all other constraints are satisfied: Each player plays exactly once each week, and all groups have the correct size. The interesting pattern of conflict positions is due to the following

Fig. 3 A "solution" for the 8−4−2 instance, conflicts highlighted

	Week 1				Week 2			
Group 1	32	24	16	8	31	23	15	7
Group 2	31	23	15	7	29	21	13	5
Group 3	30	22	14	6	27	19	11	3
Group 4	29	21	13	5	25	17	9	1
Group 5	28	20	12	4	26	22	16	12
Group 6	27	19	11	3	30	18	8	4
Group 7	26	18	10	2	32	28	14	10
Group 8	25	17	9	1	24	20	6	2

problem in the model: In clause sets (8)–(11), columns of the ladder matrix are referenced by the terms $(l \times k)$ and $(l \times k + 1)$, with $1 \le l \le w$ and $1 \le k \le g$. Clearly, each group should be assigned a distinct column in the ladder matrix, but the way in which the running variables are combined to form a column index does not guarantee that. For example, both the second group of the first week, and the first group of the second week will evaluate to column $1 \times 2 = 2 \times 1 = 2$ of the ladder matrix. As a consequence, the relevant constraints are not imposed on all groups.

To remedy this, we propose that the column index of group k in week l be uniquely determined by the expression $(l-1) \times g + k$. This yields the following revised versions of clause sets (8)–(11):

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \neg G'_{ikl} \vee \neg G'_{mkl} \vee \textsc{Ladder}_{\binom{x-i}{2}+m-i,(l-1)\times g+k}, \tag{17}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \neg G'_{ikl} \vee \neg G'_{mkl} \vee \neg\textsc{Ladder}_{\binom{x-i}{2}+m-i,(l-1)\times g+k+1}, \tag{18}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \textsc{Ladder}_{\binom{x-i}{2}+m-i,(l-1)\times g+k+1} \vee \neg\textsc{Ladder}_{\binom{x-i}{2}+m-i,(l-1)\times g+k} \vee G'_{ikl}, \tag{19}$$

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{i=1}^{x-1}\bigwedge_{m=i+1}^{x} \textsc{Ladder}_{\binom{x-i}{2}+m-i,(l-1)\times g+k+1} \vee \neg\textsc{Ladder}_{\binom{x-i}{2}+m-i,(l-1)\times g+k} \vee G'_{mkl}. \tag{20}$$

This completes the corrected formulation.

5 Improving the SAT formulation by Gent and Lynce

Having presented a correct SAT formulation for the SGP, we now look for ways to reduce computation time when solving SGP instances in practice. There are various ways to do this, for example

– reduce the number of variables
  This reduces the number of possible assignments and thus leads to a smaller search space.
– reduce the number of clauses
  Fewer clauses means fewer constraints that need to be satisfied, in effect turning more assignments into satisfying ones.
– impose symmetry breaking constraints
  This removes uninteresting solutions and can help a SAT solver to focus on more interesting parts of the search space.

We consider symmetry breaking constraints in Sect. 7, and focus for now on the first two options.

Our first observation regards the number of clauses: The revised versions of clause sets (10) and (11), namely (19) and (20), are actually not necessary: They can be omitted from the model without affecting the correctness of the formulation. The sole purpose of the ladder matrix is to reflect the fact that two players play in the same group. This means that *if* two players play in the same group, we want the ladder matrix to reflect that. If, on the other hand, a TRUE and FALSE value in the ladder matrix are adjacent, then the corresponding players need *not* necessarily play together in that group. This is due to the fact that our focus is on the players, and the ladder matrix is only an auxiliary device that we use for the specific purpose of enforcing the socialisation constraint. Other than that, the ladder matrix is of no interest, and remaining cells can assume any values. Reducing the number of clauses in this way can result in significantly less computation time.

However, one can go even further, and we now reduce the number of variables for each instance by eliminating the ladder matrix entirely. Instead of (7)–(11), we propose a different way to encode the desired socialisation constraint that lies at the core of the SGP. The set of clauses we propose can be concisely described as:

$$\bigwedge_{l=1}^{w}\bigwedge_{k=1}^{g}\bigwedge_{m=1}^{x}\bigwedge_{n=m+1}^{x}\bigwedge_{k'=1}^{g}\bigwedge_{l'=l+1}^{w}(\neg \mathrm{G}'_{mkl}\vee\neg \mathrm{G}'_{nkl})\vee(\neg \mathrm{G}'_{mk'l'}\vee\neg \mathrm{G}'_{nk'l'}). \tag{21}$$

This states the socialisation constraint in a very straight-forward manner: If two players m and n play in the same group k of a week l, then they cannot play together in any group of further weeks.

This change of the formulation makes all LADDER$_{xy}$ variables unnecessary. For each SGP instance $g{-}p{-}w$, our formulation will therefore always have exactly $\binom{g\times p}{2}\times(g\times w+1)$ fewer variables (i.e., exactly the number of ladder variables) than the formulation proposed by Gent and Lynce. The number of clauses can be less, equal, or more depending on the instance. In the next section, we assess empirically what can be gained by our change of the formulation.

6 Experimental results

We chose the two well-known instances $5{-}3{-}w$ and $8{-}4{-}w$ for benchmarking, which we consider quite representative for other instances as well.

All experiments were conducted on an Apple MacBook with a 2.16 GHz Intel Core 2 Duo CPU and 1 GB RAM. We used the two SAT solvers Walksat (Selman et al. 1993) (version 4.6) and SATO (Zhang 1997) (version 4.2). Walksat uses local search, and SATO uses the Davis-Putnam method. For SATO, we started the solver and waited with a timeout of 20 minutes. For Walksat, we tried many options and chose the cutoff in such a way that the program had about 20 minutes to solve an instance. This is a reasonable time frame to be competitive, as this is also the limit that was chosen in recently reported other approaches with similar hardware (Cotta et al. 2006).

Tables 1 and 2 show benchmark results with the (revised) SAT formulation by Gent and Lynce and our formulation, respectively. For each SGP instance, we show the number of variables and clauses of the generated SAT instance. The "Walksat" column shows the average number of seconds until a satisfying assignment was found in 10 tries. The "SATO" column shows the number of seconds until a single solution was found, averaged over 10 runs

Table 1 Revised formulation by Gent and Lynce

Instance	#Vs.	#Cl.	Walksat	SATO
5−3−1	930	6105	0.00s	0.01s
5−3−2	1755	12210	0.02s	0.02s
5−3−3	2580	18315	–	–
5−3−4	3405	24420	–	–
5−3−5	4230	30525	–	–
5−3−6	5055	36630	–	–
8−4−1	5744	52928	–	0.06s
8−4−2	10992	105856	–	0.11s
8−4−3	16240	158784	–	–
8−4−4	21488	211712	–	–
8−4−5	26736	264640	–	–
8−4−6	31984	317568	–	–

Table 2 Our formulation

Instance	#Vs.	#Cl.	Walksat	SATO
5−3−1	300	3480	0.00s	0.12s
5−3−2	600	9585	0.00s	0.01s
5−3−3	900	18315	0.00s	0.01s
5−3−4	1200	29670	0.01s	0.03s
5−3−5	1500	43650	2.42s	0.04s
5−3−6	1800	60255	98.94s	–
8−4−1	1280	33088	0.00s	0.03s
8−4−2	2560	97920	0.01s	0.10s
8−4−3	3840	194496	0.05s	1.16s
8−4−4	5120	322816	0.75s	–
8−4−5	6400	482880	0.98s	–
8−4−6	7680	674688	198.92s	–

to reduce variance of measured run times. The symbol "–" means that no solution was found within the time limit.

Despite trying many different options with Walksat, we did not find many solutions using the (revised) formulation by Gent and Lynce. In contrast, we could solve all given instances with Walksat's "novelty" option and a cutoff parameter of 10^7 by using our formulation.

It is clear from these figures that our changes to the SAT formulation can result in large performance improvements when solving SGP instances in practice with this method. A SAT-based approach towards the SGP thus seems now at least more promising than previously, when results were much further from being competitive with other approaches.

7 Symmetry breaking

Another strategy that can reduce computation time when solving SAT instances in practice is to eliminate uninteresting branches of the search tree by breaking some of the symmetries inherent to the underlying problem. Clearly, the SGP is a highly symmetric problem: Players within groups, groups within weeks, and weeks themselves can be reordered arbitrarily. Yet, schedules with different orders of players within the same group lead to distinct solutions in the SAT formulation above. Especially when using SAT solvers that use a complete backtracking algorithm, it can help to eliminate these symmetries and thus force the solver away from regions of the search space that have already been considered in some variant. Gent and Lynce propose the following set of clauses to break the symmetry among players within each group:

$$\bigwedge_{i=1}^{x}\bigwedge_{j=1}^{p}\bigwedge_{k=1}^{g}\bigwedge_{l=1}^{w}\bigwedge_{m=1}^{i-1} \neg \mathrm{G}_{ijkl} \vee \neg \mathrm{G}_{m(j+1)kl}. \tag{22}$$

Clearly, j must in fact range from 1 to $p - \mathbf{1}$. Also, since the players within each week must be distinct, m can range from 1 to $\mathbf{i}$. With these modifications, the clause set (22) ensures that the players within each group are in strictly increasing numerical order.

All groups within a single week can be ordered by their first players, for which Gent and Lynce propose the set of clauses:

$$\bigwedge_{i=1}^{x}\bigwedge_{k=1}^{g}\bigwedge_{l=1}^{w}\bigwedge_{m=1}^{i-1} \neg \mathrm{G}_{i1kl} \vee \neg \mathrm{G}_{m1(k+1)l}. \tag{23}$$

Again, k must actually range from 1 to $g - \mathbf{1}$. From (22) and (23), it follows that player 1 is the first player of the first group in each week. Using this fact, weeks can be ordered lexicographically by the *second* golfer playing in the first group of each week. Gent and Lynce propose the set of clauses:

$$\bigwedge_{i=1}^{x}\bigwedge_{k=1}^{g}\bigwedge_{l=1}^{w}\bigwedge_{m=1}^{i-1} \neg \mathrm{G}_{i2kl} \vee \neg \mathrm{G}_{m2k(l+1)}. \tag{24}$$

Again, l must actually range from 1 to $w - \mathbf{1}$. Also, this set of clauses removes more solutions than intended, as it erroneously enforces the constraint for *each* group, instead of only the first group of each week. In addition, we know that the second player in the first group must be distinct for each week, since player 1 also plays in this week, and players cannot meet more than once. We can therefore demand *strictly* ascending second players in the first group of each week, by using instead the following set of clauses:

$$\bigwedge_{i=1}^{x}\bigwedge_{l=1}^{w-1}\bigwedge_{m=1}^{i} \neg \mathrm{G}_{i21l} \vee \neg \mathrm{G}_{m21(l+1)}. \tag{25}$$

This completes the symmetry breaking constraints.

8 Experimental results again

It is well-known that symmetry breaking constraints can make it harder for local search methods to find solutions (Prestwich 2001), as they reduce the number of solutions and thus the solution density. In contrast, solution methods that involve backtracking search can greatly benefit from symmetry breaking constraints, and we thus expect a positive impact when using these constraints together with the SATO solver.

Table 3 Our formulation, including symmetry breaking constraints

Instance	#Vs.	#Cl.	Walksat	SATO
5−3−6	1800	67455	–	1.98s
5−3−7	2100	91965	–	3m5.79s
8−4−4	5120	389872	–	1.58s
8−4−5	6400	566832	–	3.64s
8−4−6	7680	775536	–	12.83s
8−4−7	8690	1015984	–	4m3.85s

We added the three symmetry breaking constraints of Sect. 7 to the model we propose in Sect. 5, in the hope to further reduce computation time and solve SGP instances that we previously could not solve within the time limit.

Table 3 shows our results for those instances that could not be solved previously. As can be seen, these results largely confirm our expectation: Adding symmetry breaking clauses has a very positive effect for the backtracking solver SATO, with which we can now solve many more instances. Kirkman's schoolgirl problem, instance 5−3−7, can now be solved, and the original SGP can be solved for up to 7 weeks.

When trying to solve instance 5−3−7 with all symmetry breaking constraints, SATO exited with the message "Bus error". Running SATO in the debugger GDB (version 6.3.50) showed a memory access error in the procedure `clean_tapes` for this instance. We then removed the last two constraints, and broke only the symmetry arising from different player orders within groups to solve this instance. All other instances could be solved with all three symmetry breaking constraints.

As expected, adding symmetry breaking constraints has very adverse effects for Walksat, which is based on local search. None of the instances could be solved within the time limit, not even 5−3−6, which could be solved before adding symmetry breaking clauses.

9 More symmetry breaking

The symmetry breaking constraints of Sect. 7 do not break all symmetries of solutions: In addition to the mentioned symmetries, players can be renumbered arbitrarily. Thus, different permutations can still yield distinct but isomorphic solutions. See Barnier and Brisset (2005) for an example.

In Frisch et al. (2002), a very different model for the SGP is proposed, which allows to break much of this symmetry: Let $M_{w\times g, g\times p}$ denote a matrix of Boolean variables. Each column corresponds to a player, and each row corresponds to one group. The Boolean value at position (i, j) denotes whether player j plays in group i. The necessary constraints on M are quite obvious. Much of the symmetry arising from player permutations can now be broken by imposing a lexicographic "less than" constraint on columns. Similarly, the symmetry among groups can be broken by imposing a lexicographic "less than" constraint on the rows corresponding to the groups of each week. Unfortunately, this still does not remove all isomorphic solutions: As shown in Flener et al. (2002), lexicographically ordering both rows and columns does not break all compositions of rows and column symmetries. As this model was also shown to suffer from overheads due to large arising vectors (Frisch et al. 2002), we do not consider it further.

It is always possible to break all symmetries of a problem with lexicographic constraints (Crawford et al. 1996). However, since a super-exponential number of constraints are necessary in general, this is often infeasible in practice.

10 Comparison with other approaches

Working with SAT formulations can be quite error-prone, but there are also several benefits of SAT-based approaches for the SGP over other methods. For example, in the case of constraint based formulations, there is no commonly used modeling language that all solvers understand, and it is thus harder to run the same formulation with different constraint solvers. In contrast, the DIMACS format is widely understood by all common SAT solvers, and one can easily apply different SAT solvers and methodologies to the same model.

Existing metaheuristic methods for the SGP (Dotú and Hentenryck 2005; Cotta et al. 2006), are quite simple and perform surprisingly well on many instances, but they are typically incomplete and can therefore not be used to show that an instance cannot be solved. With complete SAT solvers, on the other hand, this is possible, and SAT solvers employing local search can still be used with the same model.

Although all instances of the SGP that we could so far solve with SAT solvers are also solvable with other known methods, we therefore consider it worthwhile to try different SAT models and solvers in the future.

11 Conclusion and future work

We revisited an existing SAT encoding of the SGP and revised some of its clauses. We then proposed an alternative encoding of the socialisation constraint which forms the core of the SGP and related problems. Our formulation always results in fewer variables, while the number of clauses can be smaller or larger depending on the instance. Our formulation can lead to considerably improved running times when solving SGP instances with common SAT solvers.

The existence of freely available and continuously evolving SAT solvers makes it attractive to try out SAT formulations for scheduling problems such as the SGP, and several interesting questions and opportunities for future research present themselves: First, there is hope that still better problem formulations or symmetry breaking constraints exist, which could result in increased performance.

Second, the SAT formulation itself could lead to new insights into the SGP due to its simplicity. By analyzing SAT encodings of SGP instances that are known to be unsolvable, one could arrive at better bounds and hardness estimates of other instances, about which only very little is known so far.

Finally, adding *implied* constraints to the model could further reduce computation time.

References

Anderson, I. (1997). *Combinatorial designs and tournaments*. Oxford: Clarendon.

Barnier, N., & Brisset, P. (2005). Solving Kirkman's schoolgirl problem in a few seconds. *Constraints*, *10*(1), 7–21.

Colbourn, C. J. (1984). The complexity of completing partial Latin squares. *Discrete Applied Mathematics*, *8*, 25–30.

Colbourn, C. H., & Dinitz, J. H. (1996). *The CRC handbook of combinatorial designs*. Boca Raton: CRC Press.

Cotta, C., Dotú, I., Fernández, A. J., & Hentenryck, P. V. (2006). Scheduling social golfers with memetic evolutionary programming. In *LNCS: Vol. 4030. Hybrid metaheuristics* (pp. 150–161). Berlin: Springer.

Crawford, J. M., Ginsberg, M. L., Luks, E. M., & Roy, A. (1996). Symmetry-breaking predicates for search problems. In *KR* (pp. 148–159).

Dotú, I., & Hentenryck, P. V. (2005). Scheduling social golfers locally. In *LNCS: Vol. 3524. CPAIOR* (pp. 155–167). Berlin: Springer.

Flener, P., Frisch, A. M., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., & Walsh, T. (2002). Breaking row and column symmetries in matrix models. In *LNCS: Vol. 2470. CP'02*. Berlin: Springer.

Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., & Walsh, T. (2002). Global constraints for lexicographic orderings. In *LNCS: Vol. 2470. CP'02*. Berlin: Springer.

Gent, I., & Lynce, I. (2005). A SAT encoding for the social golfer problem. In *IJCAI'05 workshop on modelling and solving problems with constraints*.

Gent, I. P., & Walsh, T. (1999). CSPLib: a benchmark library for constraints. In *LNCS: Vol. 1713. CP'99*. Berlin: Springer.

Harvey, W., & Winterer, T. (2005). Solving the MOLR and social golfers problems. In *LNCS: Vol. 3709. CP'05* (pp. 286–300). Berlin: Springer.

Petrie, K. E., & Smith, B. (2004). Dynamic symmetry breaking in constraint programming and linear programming (Technical report).

Prestwich, S. (2001). First-solution search with symmetry breaking and implied constraints. In *Proceedings of the CP'01 workshop on modelling and problem formulation*.

Selman, B., Kautz, H., & Cohen, B. (1993). Local search strategies for satisfiability testing. In *Second DIMACS implementation challenge*.

Zhang, H. (1997). SATO: an efficient propositional prover. In *LNAI: Vol. 1249. CADE*. Berlin: Springer.