Extra Constraints for the Social Golfers Problem

Francisco Azevedo[1], and Hau Nguyen Van[2]

[1] CENTRIA, Universidade Nova de Lisboa, Portugal
[2] Information Technology Department, Hanoi University of Technology, Vietnam
fa@di.fct.unl.pt, nvhau66@gmail.com

Abstract. The Social Golfers Problem (SGP) has been extensively used in recent years by the Constraint Programming (CP) community for Symmetry Breaking (SB) research, with remarkable results. In this paper, in addition to static SB constraints, we present redundant constraints that drastically help further improve solving of SGPs, as shown with experimental results. In addition, we also introduce new constraints by fixing some values that, in spite of being probably incomplete, allow obtaining solutions to extra instances.

1 Introduction

Highly symmetric combinatorial problems constitute a great challenge in AI. For that, SB, namely in CP, has been the subject of much research in recent years [1,2,5,8-10]. One such problem is SGP (problem 10 in csplib.org). A g-s-w instance of SGP can be stated as follows: a group of $n=g\times s$ golfers wish to play golf each week, arranged into g groups of s golfers; find a playing schedule for w weeks such that no two golfers play together more than once. This problem has $(s!)^{gw}(g!)^{w}w!(gs)!$ symmetrical solutions since: 1) players inside groups can be exchanged; 2) groups inside weeks can be exchanged; 3) weeks can be exchanged; 4) players can be renumbered.

Time to search solutions is dependent on the size of the search tree. The key idea of this paper is to prune search tree further by adding extra constraints to the usual SB static constraints on SGP. In this paper, we describe two different types of constraints that we add to the model to solve it faster. The paper is structured as follows: we first define SGP and model it, giving numerous common constraints and describe our two techniques. Then, we present experimental results, and make some suggestions as to how the technique could be improved further.

2 Breaking Symmetries Statically and Extra Constraints

Let $G_{i,j} \subseteq \{1,2,\ldots,n\}$ represent the j^{th} group of s golfers in week i. All groups in a week are disjoint ($G_{i,j} \cap G_{i,j'} = \emptyset$) and every pair of groups from different weeks shares at most one element ($1 \le i < i' \le w,\ 1 \le j < j' \le g \quad |G_{i,j} \cap G_{i',j'}| \le 1$)

2.1 Basic SB Constraints (Basic)

Symmetry (1) above is removed by ordering golfers inside groups. Symmetry (2) is handled by ordering the groups of every week. Let $M1_{i,j}$ represent the minimum element of $G_{i,j}$. Then, for each week i, we post constraints $M1_{i,j} < M1_{i,j+1}$ (for $j < s$).

We handle symmetry (3) in roughly the same way, forcing $M1_{i,1}$ (first golfer) to be 1, and considering the second smallest element for $G_{i,1}$; we call it $M2_i$. We thus order weeks by posting constraints $M2_i < M2_{i+1}$ (for $i < w$).

Symmetry (4) is harder to handle: one needs to use advanced techniques for dynamic symmetry breaking to handle them fully [5,8,9]. Some parts of the symmetry can be removed, though, by setting some of the values statically [1,2,9].

The first week can be statically set. Each $G_{1,i}$ can be set to $\{i.g+1, \ldots, i.(g+1)\}$. For example, group $G_{1,2}$ for the *3-4-w* instance is set to {4, 5, 6}. After the first week is fixed, there are $w.(s!)^g.(g!)$ symmetries. The first golfer of the first s groups of each week can also be fixed with the smallest possible value in order (these are golfers $1\ldots s$). The first group of the second week can still be further fixed with the smallest possible players. For example, group $G_{2,1}$ for such instance is {1, 4, 7}.

2.2 Narrow Domains (ND)

We now propose new redundant constraints for this problem. Let us consider for 20 golfers, to be arranged in 5 groups of 4 in each week. In the first week we have:

{{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16},{17,18,19,20}}

In another arbitrary week we have {{1,?,?,?},{2,?,?,?},{3,?,?,?},{4,?,?,?},{?,?,?,?}}.

Let $G_{i,j}(k)$ refer to the kth element of $G_{i,j}$, when seen as an ordered sequence, as in our CLP(*FD*) model. $G_{i,2}(3)$ can not be in {17,18,19,20}; otherwise, the domain of $G_{i,2}(4)$ becomes empty ($G_{i,2}(3) < G_{i,2}(4)$). So $G_{i,2}(3)$ must be less than 17. Similarly, if $G_{i,j}(3)$ is in {5,6,7,8} then the domain of $G_{i,2}(2)$ becomes empty. So $G_{i,2}(3)$ must be greater than 8. Hence, $8 < G_{i,2}(3) < 17$. In the same way, we obtain $4 < G_{i,2}(2) < 13$ and $12 < G_{i,2}(4) < 21$. In general, we have $\boldsymbol{s*(k-1) < G_{i,j}(k) < n-s*(s-k)+1}$.

Notice that in instances of the form *g-g-w*, a variable domain is limited to g values (all values in a group of the first week). For the 4-4-*w* example instance of Table 1, $G_{3,3}(2)$ has domain {5,6,7,8}, $G_{3,3}(3)$ has domain {9,10,11,12}, and $G_{3,3}(4)$ has domain {13,14,15,16}.

Table 1. Part of a 4-4-*w* instance

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1				2				3				4			
1				2				3	?	?	?	4			

2.3 Fixing Players (F)

Additionally, we propose an additional fixing of players: the players in the last group of the first week are orderly assigned to the last position in the last groups of the second week (see Table 2). Intuitively, it is reasonable, because these players must be set at a last position and players together in a same group in the first week are placed in ordered groups in the second week. Actually, it is probably incomplete for $g < s$ (it is complete for $g=s$), but surprisingly, it is very good in practice. And of course, there is a trade-off between completeness and efficiency.

Table 2. Two first weeks of a 3-4-*w* instance

1	2	3	4	5	6	7	8	9	10	11	12
1	4	7	2	?	**10**	3	?	**11**	?	?	**12**

3 Results and Limitations

In Table 3, we present results (in seconds) to find the first solution (using SICStus Prolog [7] in Windows XP, on a Pentium III/850MHz, 256MB), using 3 different combinations of the techniques described in the previous section: **Basic**, **ND**, and **ND+F**, which uses both **ND** and **F** on top of **Basic**. Dashed entries represent aborted runs after 5 minutes. Bold entries represent the best-known number of weeks for *g*-*s* instances, using either CP or mathematical constructions [3]. Notice that in many instances, although not reaching the maximum, our solutions correspond to the best ones using CP (e.g. all instances for *g*=9).

Table 3. Experimental results for *Basic, F; ND+F*

g	s	w	Basic	ND	ND+F	g	s	w	Basic	ND	ND+F	g	s	w	Basic	ND	ND+F
4	2	2	0.02	0.01	0.01	6	2	6	0.22	0.18	1.20	8	2	11	1.29	1.23	1.11
		3	0.02	0.01	0.01			7	0.29	0.25	40.13			12	1.99	1.48	1.33
		4	0.06	0.03	0.03			8	0.39	0.32	313.78			13	2.39	1.79	1.59
		5	0.07	0.05	0.05			9	0.52	0.41	-			14	2.53	2.17	1.85
		6	0.10	0.08	0.08			10	0.63	0.52	-			**15**	**2.74**	**2.46**	**2.12**
		7	**0.12**	**0.11**	**0.10**			**11**	**0.77**	**0.63**	**1.02**		3	7	42.45	0.99	-
	3	2	0.02	0.01	0.01		3	6	-	-	0.35			8	-	1.30	-
		3	0.04	0.03	0.03			7	-	9.16	1.48			9	-	3.70	-
		4	**0.08**	**0.06**	**0.07**			**8**	-	-	-		4	4	1.25	0.45	0.40
	4	2	0.02	0.02	0.02		4	5	47.74	0.57	0.67			5	2.47	0.78	0.69
		3	0.07	0.04	0.05			6	-	-	-			6	53.37	1.61	1.06
		4	0.14	0.09	0.07		5	4	39.32	0.39	83.72			7	32.27	61.23	1.48
		5	**0.20**	**0.14**	**0.12**			5	-	-	-			8	-	-	2.16
5	2	5	0.11	0.09	0.09			**6**	-	-	-			9	-	-	2.49
		6	0.14	0.13	0.19		6	**3**	**0.38**	**0.20**	**0.22**		5	6	-	138.77	135.24
		7	0.19	0.17	0.63	7	2	7	0.36	0.33	0.33		6	5	-	231.23	229.43
		8	0.26	0.24	0.64			8	0.49	0.43	0.47		7	4	-	10.04	11.50
		9	**0,29**	**0.27**	**0.30**			9	0.63	0.56	0.60		8	**9**	**-**	**-**	**6.95**
	3	2	0.02	0.01	0.01			10	0.81	0.72	0.79	9	3	11	-	28.65	-
		3	0.03	0.05	0.04			11	1.01	0.89	0.92		4	8	-	18.91	-
		4	0.05	0.09	0.09			12	1.20	1.06	1.12		5	6	-	5.13	-
		5	0.21	0.16	0.20			**13**	**1.27**	**1.30**	**7.85**		6	5	-	6.06	-
		6	9.46	3.51	3.46		3	8	12.81	6.45	7.85		7	4	-	3.58	-
		7	**170.12**	**180.12**	**0.95**			9	-	-	-		8	3	-	0.84	0.81
	5	2	0.05	0.04	0.03		4	6	11.22	13.21	-		9	3	2.63	0.96	0.93
		3	0.15	0.10	0.09		5	5	-	1.91	-	10	2	**19**	**-**	**7.02**	**346.00**
		4	0.66	0.20	1.35		6	4	-	0.62	-		4	9	-	205.00	-
		5	1.22	0.34	1.88			5	-	-	-		9	3	-	1.30	1.24
		6	**1.79**	**0.49**	**0.78**		7	**8**	**-**	**-**	**287.30**		10	3	-	1.45	1.40

Adding **ND** constraints to **Basic** resulted in generally faster solutions and allowed us to obtain many more solutions, especially in harder instances. If we further add **F**, we do not solve as many instances but we are able to solve some of them particularly faster (e.g. 2 orders of magnitude in 5-3-7) and also solve new ones (e.g. 2 more weeks for *g-s=8-4* in a couple of seconds, and the optimum 8-8-9 in 7 seconds).

In Table 4 (where *NDF* stands for the combined best of **ND** and **F**), we compare our results with SB by Dominance Detection (SBDD) (these on a Pentium II/450 MHz) with and without merged checks (*mc*) [5]. All instances that SBDD can solve are now solved in less than 1 second with our techniques (including 5-3-7, the Kirkman's Schoolgirl problem [2]). Furthermore, we can still solve many more instances (in particular, for *g-s=6-2* we obtain 11 weeks in 0.63 seconds, whereas SBDD requires 1 hour to obtain just 6 weeks).

Table 4. Time results of SBDD for SGP in comparison with our NDF techniques

g	s	w	SBDD	SBDD mc	NDF	g	s	w	SBDD	SBDD mc	NDF
3	3	2	0.0	0.0	0.01	5	5	2	0.1	0.1	0.04
		3	0.0	0.0	0.01			3	0.2	0.2	0.10
		4	0.1	0.1	0.03			4	2277.2	999.9	0.20
4	2	2	0.0	0.0	0.01			5	-	3639.3	0.34
		3	0.1	0.1	0.01			6	-	-	0.49
		4	0.1	0.1	0.03	6	2	6	6494.6	3607.1	0.18
		5	0.2	0.2	0.05			7	-	-	0.25
		6	0.3	0.3	0.08			8	-	-	0.32
		7	0.4	0.3	0.11			9	-	-	0.41
	3	2	0.0	0.0	0.01			10	-	-	0.52
		3	0.1	0.1	0.03			11	-	-	0.63
		4	9.9	7.6	0.06		3	2	0.1	0.1	0.02
	4	2	0.0	0.0	0.02			3	0.2	0.2	0.06
		3	0.1	0.1	0.04			4	0.4	0.4	0.15
		4	0.2	0.2	0.09			5	1683.0	374.1	0.24
		5	0.3	0.3	0.14			6	-	-	0.35
5	2	5	0.3	0.3	0.09			7	-	-	1.48
		6	27.2	16.5	0.13			8	-	-	-
		7	504.4	216.1	0.17		4	2	0.2	0.1	0.04
		8	2842.4	1140.8	0.24			3	0.4	0.4	0.12
		9	5468.5	2410.9	0.27			4	0.5	0.6	0.22
	3	2	0.1	0.1	0.01			5	1461.0	537.2	0.57
		3	0.2	0.1	0.04			6	-	-	-
		4	29.9	17.8	0.09		5	2	0.2	0.2	0.05
		5	347.8	109.3	0.16			3	0.4	0.4	0.15
		6	-	-	3.46			4	1929.0	602.7	0.39
		7	**-**	**-**	0.95			5	-	-	-
	4	2	0.1	0.1	0.03			6	-	-	-
		3	0.2	0.2	0.08		6	2	0.2	0.1	0.07
		4	118.9	11.0	0.16			3	0.4	0.3	0.20
		5	1374.6	653.0	0.84						

In Table 5, we show that our results are competitive with SGLS [4] (SGP using Local Search, in C on a 3.06GHz-processor, 512MB of RAM, thus yielding roughly a 50x speedup). In fact, we even found a solution for 3 more instances, in spite of not finding a solution for, e.g., 8-3-10 in less than 10 minutes.

Table 5. Time results of SGLS for SGP in comparison with our NDF techniques

g	s	w	SGLS	NDF	g	s	w	SGLS	NDF
6	6	3	0.01	0.21	9	3	11	0.08	28.60
7	5	5	0.07	1.92		4	8	0.09	18.90
	6	4	0.09	0.64		5	6	0.09	5.13
	7	3	0.03	0.25		6	5	0.13	6.06
		8	-	287.21		7	4	0.14	3.58
8	3	10	0.23	-		8	3	0.09	0.84
	4	8	207.77	1.82		9	3	0.19	0.96
		9	-	7.36	10	4	9	0.16	205.05
	5	6	0.37	138.23		7	5	0.41	489.30
	6	5	1.21	231.41		9	3	0.21	1.34
	7	4	0.39	10.06		10	3	0.39	1.40
	8	4	360.00	501.46					
		9	-	7.36					

4 Conclusions and Further Research

In this paper, we proposed two different types of additional constraints for the SGP. The redundant narrowing domain (**ND**) constraints allowed pruning search tree further, solving hard instances faster. When compared to SBDD, our approach obtains huge improvements, often by orders of magnitude, and easily solves extra instances. Fixing particular golfers (the other technique we presented: **F**), although probably incomplete in some cases, also allowed solving some more instances.

In future research, we would like to combine our approach with local search, dynamic symmetry breaking during search, as well as with other techniques.

References

1. Azevedo, F.: An Attempt to Dynamically Break Symmetries in the Social Golfers Problem. In: Azevedo et al. (eds): Proceedings of the 11th Annual ERCIM Workshop on Constraint Programming (CSCLP'2006), (2006), 101–115
2. Barnier, N., Brisset, P.: Solving the Kirkman's Schoolgirl Problem in a Few Seconds. In: Proceedings of CP’02. Springer (2002), 477–491
3. Colbourn, C., and Dinitz: The CRC Handbook of Combinatorial Design. CRC Press, (1996)
4. Dotú, I., and Van Hentenryck, P.: Scheduling Social Golfers Locally. In: Proceedings of CP-AI-OR’05, (2005)
5. Harvey, W.: Symmetry Breaking and the Social Golfer Problem. In: Proceedings of CP’01 Workshop on Symmetries (2001)
6. Jaffar, J., and Lassez, J.-L.: Constraint Logic Programming. In: Proceedings of the 14th ACM Symposium on Principles of Programming Languages, (1987), 111–119
7. Programming Systems Group of the Swedish Institute of Computer Science, SICStus Prolog User’s Manual, (1995)
8. Puget, J.-F.: Symmetry breaking revisited. In: Proceedings of CP’02, (2002), 446–461
9. Smith, B.: Reducing symmetry in a combinatorial design problem. In: Proceedings of CPAIOR’01, (2001), 351–359
10. Walsh, T.: General Symmetry Breaking Constraints. To appear in: Proceedings of CP’06, LNCS 4204, (2006), 650–664