Scheduling Social Golfers with Memetic Evolutionary Programming

Carlos Cotta[1], Iván Dotú[2], Antonio J. Fernández[1], and Pascal Van Hentenryck[3]

[1] Dpto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain
[2] Dpto. de Ingeniería Informática, Universidad Autónoma de Madrid, Spain
[3] Brown University, Box 1910, Providence, RI 02912, USA

Abstract. The social golfer problem (SGP) has attracted significant attention in recent years because of its highly symmetrical, constrained, and combinatorial nature. Nowadays, it constitutes one of the standard benchmarks in the area of constraint programming. This paper presents the first evolutionary approach to the SGP. We propose a memetic algorithm (MA) that combines ideas from evolutionary programming and tabu search. In order to lessen the influence of the high number of symmetries present in the problem, the MA does not make use of recombination operators. The search is thus propelled by selection, mutation, and local search. In connection with the latter, we analyze the effect of baldwinian and lamarckian learning in the performance of the MA. An experimental study shows that the MA is capable of improving results reported in the literature, and supports the superiority of lamarckian strategies in this problem.

1 Introduction

The social golfer problem has attracted significant interest since it was first posted on `sci.op-research` in May 1998. It consists of scheduling $n = g \cdot s$ golfers into g groups of s players every week for w weeks so that no two golfers play in the same group more than once. The problem can be regarded as an optimization problem if for two given values for g and s, we ask for the maximum number of weeks w the golfers can play together.

It can be easily inferred from the informal definition of the SGP given before that it constitutes is a highly combinatorial, constrained, and symmetric problem. Not surprisingly, a lot of attention has been devoted to the SGP in the constraint programming community (e.g., see [1,2,3] among others). Indeed, it raises fundamentally interesting issues in modelling and symmetry breaking, and has become one of the standard benchmarks in the area. Notice in this sense that symmetry is manifold in this problem, e.g., players can be permuted within groups, groups can be ordered arbitrarily within every week, and even the weeks themselves can be permuted. Recent developments (e.g., [4]) approach the scheduling of social golfers using innovative, elegant, but also complex, symmetry-breaking schemes.

To the best of our knowledge, no evolutionary approach has been reported in the literature to handle this problem. Here, we present a memetic algorithm (MA) that is based on the hybridization of evolutionary programming and tabu search, and that constitutes the first attempt of tackling the SGP by evolutionary techniques. While deterministic techniques such as constraint programming have addressed the SGP by detecting and breaking symmetries (e.g., [5,6,7,8]), the flexibility of MAs eases the handling of these symmetries. To be precise, their influence is mostly confined to sexual-reproduction operators such as recombination. However, as shown in this work, a MA based on selection, mutation and local search still constitutes a powerful tool for optimization, capable of performing at a state-of-the-art level for this problem.

2 The Social Golfer Problem

As mentioned in previous section, the Social Golfer Problem (SGP) consists of scheduling $n = g \cdot s$ golfers into g groups of s 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 thus specified by a triplet $\langle g, s, w \rangle$. A (potentially infeasible) solution for such an instance is given by a schedule $\sigma : \mathbb{N}_g \times \mathbb{N}_w \longrightarrow 2^{\mathbb{N}_n}$, where $\mathbb{N}_i = \{1, 2, \cdots, i\}$, and $|\sigma(i,j)| = s$ for all $i \in \mathbb{N}_g$, $j \in \mathbb{N}_w$, that is, a function that on input (i,j) returns the set of s players that constitute the i-th group of the j-th week.

2.1 Modelling the SGP

There are many possible modelings for the social golfer problem, which is one of the reasons why it is so interesting. In a generalized way, this problem can be modelled as a constraint satisfaction problem (CSP) defined by the following constraints:

– A golfer plays exactly once a week, i.e.,

$$\forall p \in \mathbb{N}_n : \forall j \in \mathbb{N}_w : \exists! i \in \mathbb{N}_g : p \in \sigma(i,j). \tag{1}$$

We will use the notation $\gamma(p,j)$ to denote the index of the group in which golfer p plays during the j-th week. This constraint can be also formalized by claiming that no two groups in the same week intersect, i.e.,

$$\forall j \in \mathbb{N}_w : \forall i, i' \in \mathbb{N}_g, i \neq i' : \sigma(i,j) \cap \sigma(i',j) = \emptyset. \tag{2}$$

– No two golfers play together more than once, i.e.,

$$\forall j, j' \in \mathbb{N}_w : \forall i, i' \in \mathbb{N}_g, i \neq i' : |\sigma(i,j) \cap \sigma(i',j')| \leqslant 1. \tag{3}$$

Let $\#_\sigma(a,b)$ be the number of times golfers a and b play together in schedule σ, i.e.,

$$\#_\sigma(a,b) = \sum_{i \in \mathbb{N}_g} \sum_{j \in \mathbb{N}_w} [\{a,b\} \subseteq \sigma(i,j)], \tag{4}$$

where $[\cdot]$ is the Iverson bracket, namely [`true`]= 1 and [`false`]= 0. We define the degree of violation of a constraint a-and-b-play-together-at-most-once $v_\sigma(a,b) = \max(0, \#_\sigma(a,b) - 1)$.

In addition to the tightly constrained structure of feasible solutions, this problem is of the foremost interest due to its high degree of symmetry. Symmetries can appear in this problem because:

– Golfers are interchangeable inside groups. This means $(s!)^{g\cdot w}$ symmetries. Notice that this symmetry arises in naive formulations of the problem in which the schedule function is defined to return a *list* of golfers for each group and week, rather than a *set* of golfers.
– Groups within a week can be exchanged. This amounts to the fact that group indexes bear no absolute meaning within a week, and implies $(g!)^w$ symmetries.
– Weeks can be arbitrarily reordered, that is, given a schedule σ, we can obtain another one by simply permutating the weekly schedules. Therefore, this means $w!$ symmetries.
– Golfers can be renumbered. This means exactly $n!$ –i.e., $(g \cdot s)!$– symmetries.

As a consequence, a very naive formulation $\langle g, s, w\rangle$ has $(s!)^{g\cdot w}(g!)^w w!(gs)!$ symmetries. Observe that the symmetries grow very rapidly as the size of the problem grows, and this may be problematic for search algorithms that ignored them (they might be mislead in the case of heuristic approaches, and/or waste computational resources in the case of complete techniques).

The symmetry problem can be remedied in different ways. For instance the first kind of symmetry is implicitly removed by using sets for modelling groups, as mentioned before. As to the symmetry inside weeks, it can be removed by ordering groups using some pre-defined total order $\prec$ (e.g., the lower the smallest element in a group is, lower the group in a week is, i.e., for any week j and $i \in \mathbb{N}_{g-1}$,

$$\sigma(i,j) \prec \sigma(i+1,j) \Leftrightarrow \min(\sigma(i,j)) < \min(\sigma(i+1,j)). \tag{5}$$

Observe that with this symmetry breaking procedure, golfer 1 is always in the first group in every week. The third symmetry can be handled in roughly the same way, that is, ordering weeks with respect to the first group in each week, using for this purpose the second lowest element in the first group, i.e., let w_i and w_{i+1} two weeks ($i \in \mathbb{N}_{w-1}$); then

$$w_i \prec w_{i+1} \Leftrightarrow \min(\sigma(1,i) \setminus \{1\}) < \min(\sigma(1,i+1) \setminus \{1\}). \tag{6}$$

Finally, symmetries among golfers are harder to handle: they can be fully removed only by using advanced techniques for dynamic symmetry breaking (see [4] for more details).

2.2 Related Work

Due to the interest that the SGP has attracted in the constraint satisfaction community, it has been extensively attacked using different techniques. Here, we mention just some of the most recent advances in solving the SGP. To begin with, Harvey and Winterer [9] have proposed to construct solutions to the SGP by using sets of mutually orthogonal latin squares. Also, Gent and Lynce [10] have recently introduced a satisfiability (SAT) encoding for the SGP. Barnier and Brisset [4] have presented a combination of techniques to efficiently find solutions to a specific instance of SGP, the Kirkman's schoolgirl problem. Global constraints for lexicographic ordering have been proposed by Frisch *et al.* [11], being used for breaking symmetries in the SGP. Also, a tabu-based local search algorithm for the SGP is described by Dotú and Van Hentenryck [12].

The SGP problem also admits a number of possible variants; for instance finding a w-week schedule with "maximum socialization" (i.e., as few repeated pairs as possible), or finding a schedule of minimum length such that each golfer plays with every other golfer at least once ("full socialization") [13]. In either case, and to the best of our knowledge, no evolutionary approach has been reported in the literature to handle this problem in any of these variants. Next section tackles this gap.

3 A Memetic Approach to the Social Golfer Problem

The application of standard population-based metaheuristics to the SGP is, if not thwarted, at least challenged by the presence of the manifold symmetries detailed in the previous section. To be precise, these symmetries are specifically relevant with respect to the performance of recombination operators (and generalizations thereof, that is, any reproductive operator constructing new solutions on the basis of two or more *parents*). If these symmetries are not implicitly broken by means of a wise representation of solutions, or the operators are not explicitly designed to take them into account, recombination attempts are doomed to fail: two similar solutions (even two identical solutions) can be considered as completely different solutions due to misregarded symmetries. As a consequence, it cannot be expected in general that the relevant features of these solutions (i.e., those information pieces ultimately responsible for the quality of the solutions) be processed in an adequate way. In this scenario, recombination is likely to behave as a highly disruptive, macromutation process. It turns out that this is precisely the general interpretation that is made of recombination in the realm of Evolutionary Programming (EP) [14]. For this reason, we have chosen an EP model as the base of our memetic approach, as shown next.

3.1 General Algorithmic Model

Following the philosophy of EP, our MA only uses mutation as the primary means to diversify the search. This relieves the need for performing symmetry breakages, and subsequently allows a simpler representation of solutions. Let σ be a $w-$week assignment for g groups of s players each, as described in Sect. 2. This assignment is encoded as a string $u = t_{11} :: t_{12} :: \cdots :: t_{1g} :: \cdots :: t_{wg}$, where $t_{ij} \in \mathbb{N}^s_{g \cdot s}$ is a permutation of the elements in the set $\sigma(i, j)$, and the operator

```
1:  for $i \in [1 : popsize]$ do
2:      let $pop[i] \leftarrow$ GenerateSolution($g, s, w$)
3:      let $f[i] \leftarrow$ ViolatedConstraints($pop[i]$)
4:  endfor
5   let $iter \leftarrow 0$
6:  do
7:      let $u \leftarrow$ Select($pop$,$f$)
8:      let $u' \leftarrow$ Mutate(u)
9:      let $(u'', f') \leftarrow$ Learning($u'$)
10:     let $i \leftarrow \max^{-1}_{i\in\{1,popsize\}}(f[i])$
11:     let $(pop[i], f[i]) \leftarrow (u'', f')$
12:     let $iter \leftarrow iter + 1$
13: until TerminationCriterion($pop$,$f$,$iter$)
14: return $pop\left[\min^{-1}_{i\in\{1,popsize\}}(f[i])\right]$
```

Fig. 1. Pseudocode of the memetic EP approach

:: indicates string concatenation. Conversely, a string $u \in \mathbb{N}_{g\cdot s}^{s\cdot g\cdot w}$ is decoded into an assignment σ by dividing it into $s-$element chunks, taking the elements in each of them as the components of a set $\sigma(i, j)$, $i \in \mathbb{N}_g$, $j \in \mathbb{N}_w$. Although no symmetries are considered in this encoding, we do have considered a basic constraint in it, namely the fact that a certain golfer cannot be scheduled into two different groups in the same week. To do so, strings are initially generated from $\mathbb{P}_{g\cdot s}^{w}$, that is, as the concatenation of w permutations of the elements in $\{1, \cdots, g \cdot s\}$. This structure of solutions is respected by all operators involved in the algorithm, whose overall pseudocode is shown in Fig. 1.

As it can be seen, our MA follows a steady-state evolution model, in which a single solution is selected, mutated, and subjected to a learning process. Regarding mutation, it is done by selecting two players from different groups in the same week, and switching their positions. The set of possible swaps is then

$$\mathcal{S}(\sigma) = \{(\langle w, p_1\rangle, \langle w, p_2\rangle) \mid \gamma(p_1, w) \neq \gamma(p_2, w)\}. \tag{7}$$

Each selected solution is subjected to a number of swaps that is Poisson-distributed with parameter $(g\cdot s)^{-1}$, that is, w swaps are performed on average. This procedure is respectful with the permutational structure of solutions, as mentioned before. As to the learning (i.e., improvement) process, it is done by means of an embedded tabu-search procedure (described in next subsection). Two strategies for conducting the learning have been considered: baldwinian and lamarckian. Firstly explored by Hinton and Nolan [15], baldwinian learning consist of performing a local improvement procedure, retaining the fitness value thus obtained, but discarding the phenotypical changes discovered. In some sense, this amounts to evaluating a solution on the basis of how good it could become, and bears some resemblance to natural evolution in that traits acquired during one's lifetime are not transmitted to the offspring. On the contrary, lamarckian learning does keep the improved phenotype as well, and injects the changes back to the genotype. This kind of learning is akin to cultural (i.e., memetic) evolution, and provides faster convergence rates (e.g., see [16]). Nevertheless, it may be also prone to premature convergence to suboptimal solutions in some cases. Determining whether this is the case in the SGP has been one of the issues considered in the experimentation.

3.2 The Tabu Search Strategy

The local improvement strategy is based on the tabu-search (TS) template, and explores the neighborhood arising from swapping golfers from different groups in the same week. This is the same neighborhood used in individual mutations, but notice that the latter consists of the iterated application of a number of swaps; hence, mutation represents a long jump in the search space, as regarded by the TS algorithm. Furthermore, from the point of view of TS, it is more effective to restrict attention just to swaps involving at least one golfer in conflict with another golfer in the same group. This ensures that the algorithm focuses on swaps which may decrease the number of violations. More formally, a pair $\langle w, p\rangle$ is said to be in conflict in schedule σ (denoted by $\upsilon_\sigma(\langle w, p\rangle)$ =`true`), if

$$\exists p' \in \sigma(\gamma(p, w), w), p' \neq p : \upsilon_\sigma(p, p') > 1. \tag{8}$$

With this restriction in mind, the set of swaps $\mathcal{S}^-(\sigma)$ considered for a schedule σ becomes

$$\mathcal{S}^-(\sigma) = \{(\langle w, p_1\rangle, \langle w, p_2\rangle) \in S(\sigma) \;\mid\; \upsilon_\sigma(\langle w, p_1\rangle)\}. \tag{9}$$

The tabu component of the algorithm is based on three main ideas. First, the tabu list is distributed across the various weeks, which is natural since the swaps only consider golfers in the same week. The tabu component thus consists of an array *tabu*, where *tabu*$[w]$ represents the tabu list associated with week w. Second, for a given week w, the tabu list maintains triplets $\langle a, b, i\rangle$, where a and b are two golfers, and i represents the first iteration where golfers a and b can be swapped again in week w. Third, the tabu tenure, i.e., the time a pair of golfers (a, b) stays in the list, is dynamic: it is randomly generated in the interval $[4, 100]$. In other words, each time a pair of golfers (a, b) is swapped, a random value ρ is drawn uniformly from the interval $[4, 100]$ and the pair (a, b) is tabu for the next ρ iterations. As a consequence, for schedule σ and iteration k, the neighborhood consists of the set of moves $\mathcal{S}^t(\sigma, k)$ defined as

$$\mathcal{S}^t(\sigma, k) = \{(\langle w, p_1\rangle, \langle w, p_2\rangle) \in \mathcal{S}^-(\sigma) \mid \nexists k' > k : \langle p_1, p_2, k'\rangle \in tabu[w]\}. \tag{10}$$

In addition to the non-tabu moves, the neighborhood also considers moves that improve the best solution found so far, i.e., the set $\mathcal{S}^*(\sigma, \sigma^*)$ defined as

$$\mathcal{S}^*(\sigma, \sigma^*) = \{(t_1, t_2) \in \mathcal{S}^-(\sigma) \mid f(\sigma[t_1 \leftrightarrow t_2]) < f(\sigma^*)\}, \tag{11}$$

where $\sigma[(w, p_1) \leftrightarrow (w, p_2)]$ denotes the schedule σ where golfers p_1 and p_2 switch their groups in week w, and σ^* denotes the best solution found so far. Observe

```
 1: for $i \in [1:w]$ do let $tabu[i] \leftarrow \emptyset$ endfor
 2: let $\sigma^* \leftarrow \sigma$
 3: let $k \leftarrow 0$
 4: while $(k \leqslant maxIter) \wedge [f(\sigma) > 0]$ do
 5:     let $f^* \leftarrow \infty$
 6:     for $(t_1, t_2) \in \mathcal{S}^t(\sigma, k) \cup \mathcal{S}^*(\sigma, \sigma^*)$ do
 7:         let $f' \leftarrow f(\sigma[t_1 \leftrightarrow t_2])$
 8:         if $f' < f^*$ then
 9:             let $(t_1^*, t_2^*) \leftarrow (t_1, t_2)$; let $f^* \leftarrow f'$
10:         endif
11:     endfor
12:     let $\tau \leftarrow$ RANDOM([4,100])
13:     let $tabu[\omega(t_1^*)] \leftarrow tabu[\omega(t_1^*)] \cup \{\langle \pi(t_1^*), \pi(t_2^*), k + \tau \rangle\}$
14:     let $\sigma \leftarrow \sigma[t_1^* \leftrightarrow t_2^*]$
15:     if $f^* < f(\sigma^*)$ then
16:         let $\sigma^* \leftarrow \sigma$
17:     endif
18:     let $k \leftarrow k + 1$
19: endwhile
20: return $(\sigma^*, f(\sigma^*))$
```

Fig. 2. Pseudocode of the tabu-search component of the MA

that the expression $f(\sigma[t_1 \leftrightarrow t_2])$ represents the number of violations obtained after performing the corresponding swap.

With the so-defined neighborhoods, the TS algorithm is described in Fig. 2. The core of the algorithm is given in lines 4-19, where local moves are iterated for a maximum number of iterations or until a solution is found. The local move is selected in line 9. The key idea is to select the best swaps in the neighborhood $\mathcal{S}^t(\sigma, k) \cup \mathcal{S}^*(\sigma, \sigma^*)$, i.e., the non-tabu swaps and those improving the best schedule. The tabu list is updated in line 13, where $\omega(\langle w, p \rangle) = w$, and $\pi(\langle w, p \rangle) = p$. The algorithm returns the best solution found and its quality.

4 Experimental Results

The experiments have been done with the steady-state MA described in previous section, using a population size of 25 individuals, binary tournament selection, and a maximum number of 2,500 evaluations. Each invocation to the TS algorithm uses $maxIter = g \cdot w$, so that there exists the possibility that the solution resulting from learning had no group in common with the original one. For each pair (g, s), we have performed series of 20 runs per increasing values of w, to determine the maximum number of weeks for which a feasible schedule can be found. The experiments have been done both with the lamarckian and the baldwinian variants of the MA.

The results are shown in Fig. 3 and Fig. 4. For comparison purposes, we include the results reported in [12] corresponding to the stand-alone application

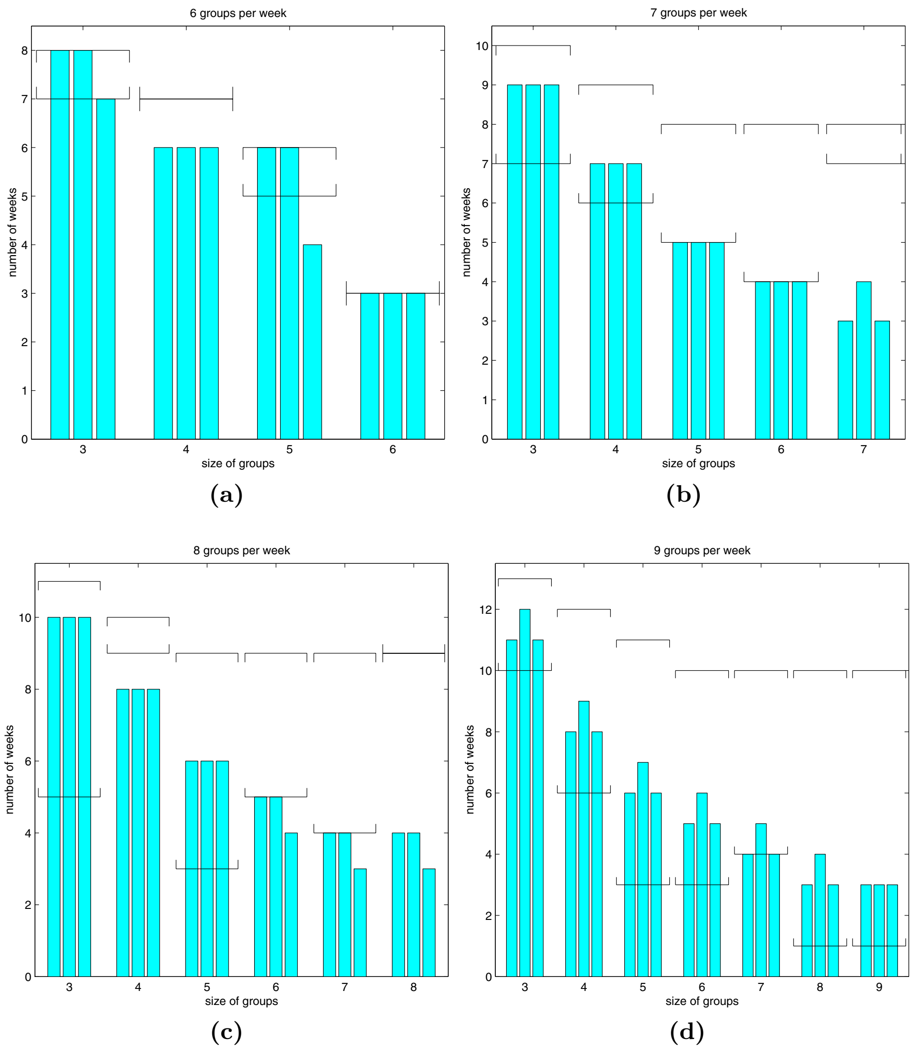

Fig. 3. Results for six groups per week (a), seven groups per week (b), eight groups per week (c), and nine groups per week (d). Each group of three bars represent (from left to right) the maximum number of weeks solved by stand-alone TS, by the lamarckian MA, and by the baldwinian MA. The horizontal brackets indicate the lower and upper bounds for the corresponding instances.

of an extended version of the TS strategy used within our MA, incorporating reinitialization mechanisms. We also indicate the upper and lower bounds for the corresponding problem instances, as reported in [17]. Notice that many of these lower bounds (i.e., best known solutions) are actually superseded by the plain application of TS. Moreover, the lamarckian MA is capable of further improving these solutions, providing better solutions for 12 problem instances,

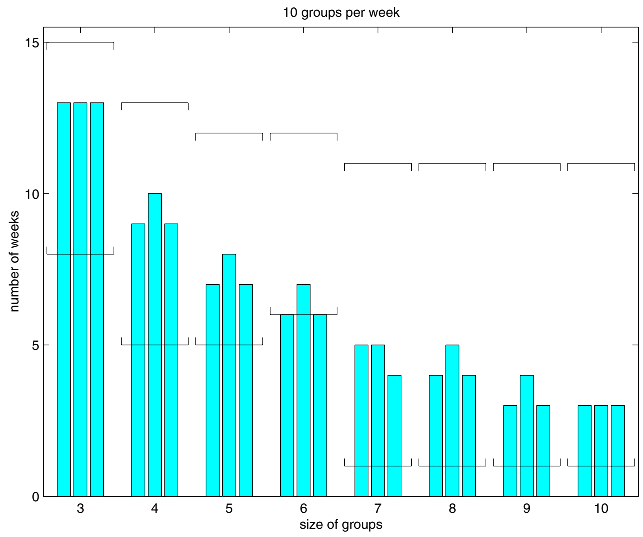

Fig. 4. Results for ten groups per week. Each group of three bars represent (from left to right) the maximum number of weeks solved by stand-alone TS, by the lamarckian MA, and by the baldwinian MA. The horizontal brackets indicate the lower and upper bounds for the corresponding instances.

and achieving the same results of TS in the remaining ones. Notice also that the performance of the baldwinian MA is markedly inferior to that of its lamarckian counterpart. Several factors may be responsible for this behavior. On one hand, the search space can have a very complex topology, thus making the baldwinian information harder to use (the improved solution found by the TS component may involve a convoluted path in the search space, very difficult to trace by means of mutation and selection). On the other hand, and related to the previous point, longer run times may be necessary for the baldwinian MA to achieve the performance level of the lamarckian MA (in these experiments, each run took from a few seconds up to around twenty minutes –in a P4 3GHz 1GB winXP computer– for the larger instances). Then again, this indicates the superiority of the latter strategy in this problem.

Further details on the performance of the two MA variants are provided in Table 1. The data correspond to the average results in the largest problem instances they could solve, thus offering a glimpse of their behavior at the edge of solvability. Obviously, these limits were shown to be farther for the lamarckian MA in Fig. 3 and Fig. 4, and hence the tables must be studied with caution. Although entries do not always correspond to homogeneous instance sizes (number of weeks in this case), we do know that the success rate for the next larger instance size is 0%. If we couple this fact with the observation that the lower part of Table 1 (i.e., the baldwinian MA) has a larger number of 100%-success entries than the upper part, we can conclude that the lamarckian MA does not simply perform better, but it also has a more gradual decline in performance

Table 1. Detailed results of the lamarckian MA (top) and the baldwinian MA (bottom) in the largest solved instance for each combination of s (size of groups) and g (groups per week). Each triplet of numbers indicate from left to right the success rate in $n = 20$ runs, the mean number of violated constraints in the best solutions found in these runs, and the standard error of the mean ($\sigma/\sqrt{n}$).

Lamarckian MA

	groups per week (g)				
size (s)	6	7	8	9	10
3	.25 .75 .10	1.00 .00 .00	1.00 .00 .00	.10 1.75 .14	1.00 .00 .00
4	.20 1.05 .15	.50 .65 .16	.50 .65 .16	.70 .50 .18	.75 .45 .18
5	.10 5.90 .48	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00
6	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	.05 2.80 .19
7	–	.05 1.95 .11	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00
8	–	–	.05 2.50 .19	1.00 .00 .00	.10 2.15 .19
9	–	–	–	1.00 .00 .00	1.00 .00 .00
10	–	–	–	–	1.00 .00 .00

Baldwinian MA

	groups per week (g)				
size (s)	6	7	8	9	10
3	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	.95 .05 .05
4	.05 1.70 .12	.05 2.00 .14	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00
5	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00
6	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	.10 1.05 .11
7	–	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00	1.00 .00 .00
8	–	–	1.00 .00 .00	1.00 .00 .00	.50 .50 .11
9	–	–	–	1.00 .00 .00	1.00 .00 .00
10	–	–	–	–	.15 2.40 .30

for increasing instance sizes. Therefore, it seems to be more scalable (or at least less sensitive to the curse of dimensionality) than the baldwinian MA. The latter exhibits an abrupt performance drop from full solvability capacity to null such capacity.

5 Conclusions and Future Work

We have presented here the first evolutionary approach to the Social Golfer Problem. Combining ideas from the realm of evolutionary programming and tabu search, we have devised a memetic algorithm capable of improving results reported in the literature. In this sense, we believe that the incorporation of intensification mechanisms such as *ad hoc* local searchers is essential to tackle this problem. Indeed, given the fact that a less-intensive strategy based in baldwinian learning performs worse than a pure lamarckian version, we hypothesize that lesser-intensive algorithms based on unbiased variation plus selection are not adequate for this problem either.

As mentioned in previous sections, symmetries play a major role in this problem. Although we have opted for not using recombination mechanisms, hence diminishing the impact of these symmetries, their consideration is an important line for future developments. We intend to approach the breakage of symmetries by smart representations and/or by problem-aware recombination operators, subsequently examining whether this symmetry-free approach results in a significant performance change.

We also plan to introduce further problem knowledge in other components of the algorithm. In this sense, Dotú and Van Hentenryck [12] have devised a constructive approach that can be shown to provide feasible solutions for certain values of w when g and s are equal and odd. In other cases, this constructive heuristic can provide a good starting point for local search. The overall results of TS endowed with this constructive heuristic are still similar to those of the lamarckian MA (despite the latter starts from a purely random initial population). Injecting the solutions provided by this constructive heuristic into the initial population may boost the performance of the MA. This issue will be dealt in the future as well.

Acknowledgements

This work was partially supported by Spanish MCyT under contracts TIN2004-7943-C04-01 and TIN2005-08818-C04-01.

References

1. Fahle, T., Schamberger, S., Sellmann, M.: Symmetry breaking. In Walsh, T., ed.: 7th International Conference on Principles and Practice of Constraint Programming. Volume 2239 of Lecture Notes in Computer Science., Paphos, Cyprus, Springer (2001) 93–107
2. Smith, B.M.: Reducing symmetry in a combinatorial design problem. In: Third International Workshop on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. (2001) 351–359
3. Sellmann, M., Harvey, W.: Heuristic constraint propagation. In Hentenryck, P.V., ed.: 8th International Conference on Principles and Practice of Constraint Programming. Volume 2470 of Lecture Notes in Computer Science., Ithaca, NY, USA, Springer (2002) 738–743
4. Barnier, N., Brisset, P.: Solving kirkman's schoolgirl problem in a few seconds. Constraints **10** (2005) 7–21
5. Ramani, A., Markov, I.: Automatically exploiting symmetries in constraint programming. In Faltings, B., Petcu, A., Fages, F., Rossi, F., eds.: Recent Advances in Constraints, Joint ERCIM/CoLogNet International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2004. Volume 3419 of Lecture Notes in Computer Science., Lausanne, Switzerland, Springer (2005) 98–112 Revised Selected and Invited Papers.

6. Prestwich, S., Roli, A.: Symmetry breaking and local search spaces. In Barták, R., Milano, M., eds.: Second International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Volume 3524 of Lecture Notes in Computer Science., Prague, Czech Republic, Springer (2005) 273–287
7. Mancini, T., Cadoli, M.: Detecting and breaking symmetries by reasoning on problem specifications. In Zucker, J.D., Saitta, L., eds.: International Symposium on Abstraction, Reformulation and Approximation (SARA 2005). Volume 3607 of Lecture Notes in Computer Science., Airth Castle, Scotland, UK, Springer (2005) 165–181
8. Sellmann, M., Hentenryck, P.V.: Structural symmetry breaking. In Kaelbling, L.P., Saffiotti, A., eds.: Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, Professional Book Center (2005) 298–303
9. Harvey, W., Winterer, T.: Solving the MOLR and social golfers problems. In van Beek, P., ed.: 11th International Conference on Principles and Practice of Constraint Programming. Volume 3709 of Lecture Notes in Computer Science., Sitges, Spain, Springer (2005) 286–300
10. Gent, I., Lynce, I.: A SAT encoding for the social golfer problem. In: IJCAI'05 workshop on Modelling and Solving Problems with Constraints, Edinburgh, Scotland (2005)
11. Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Global constraints for lexicographic orderings. In Hentenryck, P.V., ed.: 8th International Conference on Principles and Practice of Constraint Programming. Volume 2470 of Lecture Notes in Computer Science., Ithaca, NY, USA, Springer (2002) 93–108
12. Dotú, I., Hentenryck, P.V.: Scheduling social golfers locally. In Barták, R., Milano, M., eds.: International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems 2005. Volume 3524 of Lecture Notes in Computer Science., Prague, Czech Republic, Springer-Verlag (2005) 155–167
13. Gent, I., Walsh, T.: CSPLIB: A benchmark library for constraints. In Jaffar, J., ed.: 5th International Conference on Principles and Practice of Constraint Programming (CP'99). Volume 1713 of Lecture Notes in Computer Science., Alexandria, Virginia, USA, Springer (1999) 480–481
14. Fogel, L., Owens, A., Walsh, M.: Artificial Intelligence Through Simulated Evolution. Wiley, New York NY (1966)
15. Hinton, G., Nolan, S.: How learning can guide evolution. Complex Systems **1** (1987) 495–?502
16. Whitley, D., Gordon, S., Mathias, K.: Lamarckian evolution, the baldwin effect and function optimization. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature III. Volume 866 of Lecture Notes in Computer Science., Springer-Verlag (1994) 6–?15
17. Sellmann, M.: The social golfer problem. (Web site available at `http://www.cs.brown.edu/people/sello/golf.html`)