

New Heuristics for Morpion Solitaire

Heikki Hyyrö and Timo Poranen
Department of Computer Sciences
University of Tampere
Kanslerinrinne 1
FIN-33014 University of Tampere
Finland

25th October 2007

Abstract

Morpion Solitaire is a popular paper-and-pencil game. We investigate random sampling and simulated annealing algorithms for disjoint and touching models of the game. Our experiments show that simulated annealing (SA) works quite well and fast. We used it to find the best known results for non-trivial cases of the disjoint model.

1 Introduction

Morpion Solitaire is a traditional paper-and-pencil game played among children (during boring lessons and breaks) in the elementary school. It is very popular in Europe, especially in France and Belgium. The game also has a two-player version, which is introduced in a book written by Joris [7] under the name “Connector”.

Morpion solitaire is played on a square grid (e.g. squared paper) that has no explicit size limitations. Points can be drawn on the grid intersections. The standard form of the game has an initial point configuration with a Malta Cross shape, where each side of the cross has four points (see Fig. 1). The goal of the game is to make as many moves as possible. Each move consists of adding a point to an unused grid intersection and then drawing a legal line segment that connects exactly five horizontally, vertically or diagonally consecutive points. The line segment must pass through the new point. Demaine et al. [4] defined two different versions for the game: *disjoint* and *touching* models. In the disjoint model, line segments with the same direction cannot share points. In the touching model, two consecutive segments may share a common endpoint (a 1-point overlap). The game ends when no further moves are possible. A sample game with five moves is depicted in Figure 1. The last move is allowed only in the touching model.

The second author has played in Finland yet another model of the game, where it is not required that the added line segment contains the new point. If there already exist five consecutive and available points in the current game position, the player may connect those five points and place a new point in any free grid intersection. But we do not study this model in the present work.

Touching model is probably the most common version of the game. According to [4], the first known reference for it is from 1982. The best published solutions are 170 for the touching model [1] and 78 and for the disjoint model [11]. The high score of the touching model has been obtained without computational tools by using only paper and pencil. Disjoint model's high score was 68 (constructed by paper and pencil) until 2006 [11], when the preliminary version of our algorithm improved the score upto 74. After that, Cazenave's [3] reflexive Monte-Carlo search algorithm improved the solution upto 78. Finally, our simulated annealing algorithm obtained the current high score 79.

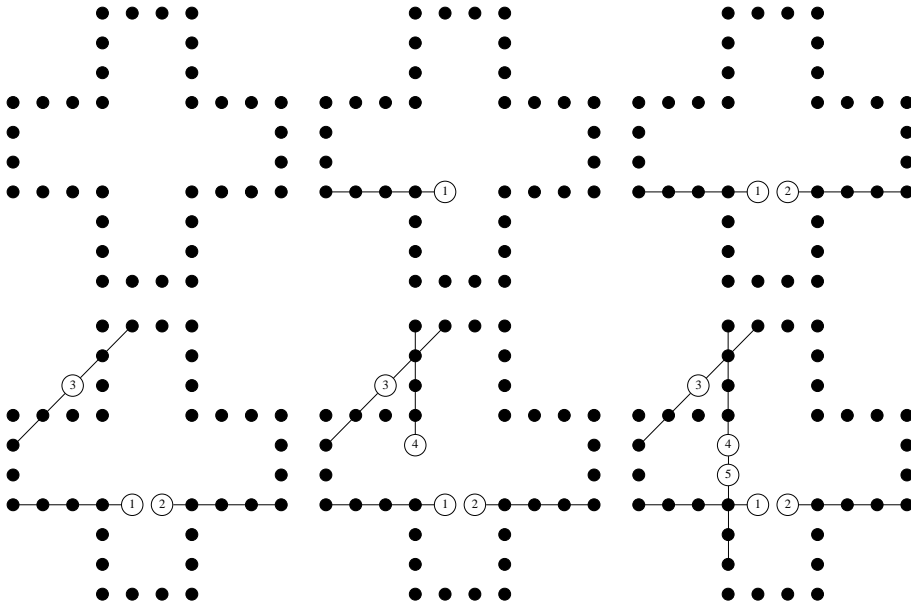


Figure 1: The initial configuration A_4 and five example moves. The last move is allowed only in the touching model. The grid lines are not shown to clarify the presentation.

Morpion solitaire has been studied in a few publications. The disjoint model has been used as a test case for an evolutionary algorithm by Juillé [8], obtaining originally a solution of 117 and later [9] 122 moves. It is reported in [1] that this method has later been improved by Pascal Zimmer to reach 143 moves. Helmstetter and Cazenave [6] have done a computational study of transpositions in the move sequences, and Flammenkamp [5] has proposed upper bounds for the disjoint model. Recently Demaine et al. [4] have discussed more general versions of the game and provided new upper and lower bounds for several of these. They also showed that in its most general form the game is NP-hard and the high score is inapproximable within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $P = NP$.

In this work we adopt the same notations as used in [4]. They generalized standard Morpion solitaire to any positive *thickness*. Under thickness k , the game uses an initial Malta Cross -shaped point configuration with side-length k and requires that each added line-segment connects exactly $k + 1$ consecutive points. Hence the original game has thickness 4. The initial Malta Cross

configuration for thickness k is denoted by A_k , and the maximum number of moves attainable from A_k is denoted by $G_k(A_k)$ under the disjoint model and by $G'_k(A_k)$ under the touching model. The initial configuration A_4 (the original game) is illustrated in Figure 1.

For original Morpion solitaire, it is previously known that $78 \leq G_4(A_4) \leq 141$ and $170 \leq G'_4(A_4) \leq 704$ [4]. Flammenkamp [5] has claimed an upper bound $G'_4(A_4) \leq 324$, but the validity of the proof seems unclear [4]. Demaine et al. [4] demonstrated that $G_1(A_1) = G'_1(A_1) = \infty$, $G_2(A_2) = G'_2(A_2) = \infty$, and $G_k(A_k) = G'_k(A_k) = 12$ for $k \geq 5$. For $k = 3$ they presented the bounds $31 \leq G_3(A_3) \leq 48$, and $56 \leq G'_3(A_3) \leq 192$.

In this work we examine two new heuristic approaches for the Morpion Solitaire. We investigate Monte Carlo -type random sampling (RS) and simulated annealing (SA) meta-heuristics and show that they can be used to obtain good solutions for different models of the game. We have been able to improve the best known lower bounds (and thus game records) in three out of the four interesting variants of the game. The results are that $G_4(A_4) \geq 79$, $G_3(A_3) \geq 35$ and $G'_3(A_3) \geq 62$. Our SA-based method is relatively competitive also in the one remaining interesting case of touching model with $k = 4$. Although our result 143 in that case is quite far from the manually constructed record $G'_4(A_4) \geq 170$, it is comparable to previous computationally generated records of 122 [9], 136 [6] and 143 [1].

In the next section we introduce our random sampling (RS) and simulated annealing (SA) optimization schemes for Morpion solitaire. Our computational experiments are described and reported in Section 3. Conclusions and future work directions are discussed in Section 4.

2 Heuristics for Morpion solitaire

In this section we propose heuristic solutions for Morpion solitaire. The basic setting is as follows. We are given a current game configuration P , which can be expressed as a set of moves. A single move may be described by the location of the added point and the orientation and alignment of the corresponding line. Our methods employ probabilistic local search with respect to a single-move-neighborhood of P . That is, they evolve solutions incrementally by moving from configuration P to a configuration P' , where P and P' differ by a single move.

2.1 Random sampling

The first method we investigated uses random sampling (RS) in order to identify (hopefully) good moves. We have tried two variants of this scheme. Given a current game position P and a parameter s that tells the sampling size, the two variants select the next move as described below.

The first variant (RS1) considers each currently possible move, one at a time. Each move is scored by computing the average number of moves made in s random endgames. Each such endgame first makes the currently evaluated move and then continues by making randomly selected moves until no more moves are possible. The move with the highest average endgame score will be performed, and the sampling process is repeated all over again if there are moves left.

In a given game position P , the second variant (RS2) plays s completely random endgames. That is, also the first move of the endgame is selected randomly. Afterwards the first move of the best of these s endgames is selected as the best move. The selected move is then performed, and the sampling process is repeated again as long as there are moves left.

We note that for a fixed s , the first variant is expected to make roughly x times more moves than the second, where x is the average number of moves available at a given game position.

This type of methods have been applied, for instance, in algorithms for the well-known two-player game “Go” [2].

Algorithms 2.1 and 2.2 show pseudocodes for the two random sampling variants RS1 and RS2, respectively. The initial configuration is not explicitly mentioned, but throughout this work we use only A_3 or A_4 .

RS1(s)

```

1  for the current position  $P$ , construct a set  $F = \{f_1, f_2, \dots, f_n\}$  of forward moves.
2  if  $|F| > 0$ 
3      then for  $i \leftarrow 1$  to  $n$ 
4          do play  $s$  random endgames from the position  $P'_i = P \cup f_i$  and
              calculate average score  $ave(f_i)$  of the endgames.
5          set  $P \leftarrow P \cup f_j$ , where  $ave(f_j) \geq ave(f_i)$  for all  $i \in \{1, 2, \dots, n\}$ .
6          go to Step 1;
7  else return the best found result;
```

Algorithm 2.1: Random sampling variant RS1 for Morpion Solitaire.

In Step 2 of Algorithm 2.1, a set of *forward moves*, $F = \{f_1, f_2, \dots, f_n\}$, is constructed. F consists simply of the possible moves from current position P . RS1 then plays s random endgames for each move $f_i \in F$, ie. starting from position $P' = P \cup f_i$, and records the average score $ave(f_i)$. After this a move f_j with a maximal average score is selected, resulting in the update $P \leftarrow P \cup f_j$ of the current game position. The whole process is repeated until there are no more moves (ie. F is empty after step 2). Finally the best solution encountered during the process is returned. This best result can correspond to either the end position or any of the sample-endgames played during the process.

The pseudocode given for RS2 in Algorithm 2.2 uses similar individual steps and notation as Algorithm 2.1. Also RS2 takes into account all sample-endgames in reporting the best found result.

2.2 Simulated annealing

The main principles of simulated annealing (SA) were first introduced in an article by Metropolis et al. [12] and later generalised by Kirkpatrick, Gelatt and Vecchi [10]. The main difference between RS and SA is that in SA we can do also backward moves during the run of the algorithm. This makes it possible to avoid getting stuck in locally optimal solutions without having to restart the search from scratch.

RS2(s)

```
1 for the current position  $P$ , construct a set  $F = \{f_1, f_2, \dots, f_n\}$  of forward moves.
2 if  $|F| > 0$ 
3   then for  $i \leftarrow 1$  to  $s$ 
4     do select a random move  $f_h \in F$ , play a random endgame from
        position  $P'_i = P \cup f_h$  and record the result  $res(f_h)$  of the endgame.
5     set  $P \leftarrow P \cup f_j$ , where  $res(f_j) \geq res(f_i)$  for all  $i \in \{1, 2, \dots, s\}$ .
6     go to Step 1;
7   else return the best found result;
```

Algorithm 2.2: Random sampling variant RS2 for Morpion Solitaire.

Given a current configuration P , our simulated annealing method for Morpion solitaire performs a step in the following manner.

Now the move-neighborhood of P consists of two separate sets, F and B . As with random sampling, F is the set of possible forward moves. B is the set of possible backward moves. A move $f_i \in P$ belongs to B if and only if removing f_i from P does not cause any other move $f_j \in P$ to become invalid. This condition can be checked for a move f_i by counting how many lines are connected to the corresponding grid point: $f_i \in B$ if and only if its grid point is connected only to its own line. For example the game position shown in Figure 3 has three backward moves (the ones numbered 56, 63, 64, 65, 79), and no forward moves.

First we select a move m randomly from the set $F \cup B$. If $m \in F$, the move is performed immediately. This results in the new game position $P \cup f$. But if $m \in B$, we perform a further randomized test that depends on the current temperature.

If the test outcome is positive or $F = \emptyset$, the backward move $m \in B$ is performed and the new game position becomes $P \setminus f$. Otherwise, if the test outcome is negative, we select and perform a random forward move $m' \in F$, and the new game position is $P \cup m'$.

The search process is regulated by four parameters, which are the initial temperature t_0 , the frozen temperature t_1 , the cooling ratio α , and the number of iterations r . The stepwise search for better solutions begins with initial temperature $t = t_0$, and ends when t has been cooled below the frozen temperature t_1 . For each temperature value t , the search performs r moves and then cools the temperature by setting $t \leftarrow \alpha t$, where it is required that $0 < \alpha < 1$.

The probability of accepting backward moves decreases with temperature. The traditional choice with SA is to use a Sigmoid-type function in determining this probability. We use the function $e^{-1/t}$.

Algorithm 3.1 shows pseudocode for our SA for Morpion solitaire.

3 Results

We have conducted experiments with the three discussed heuristic methods RS1, RS2 and SA. The methods were implemented in C programming language. We

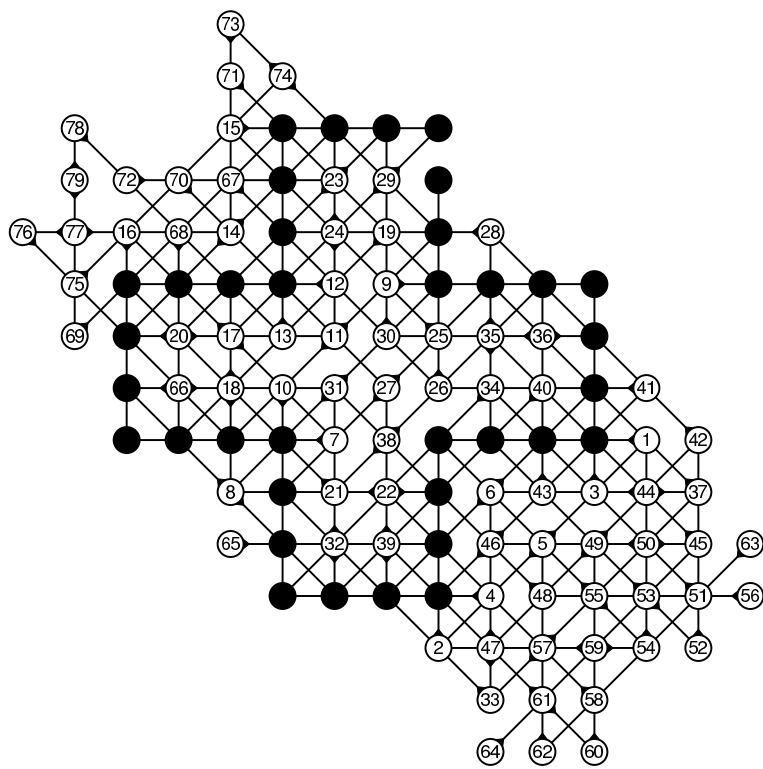


Figure 3: $G_4(A_4) \geq 79$.

paid reasonable attention to the efficiency of the implementations. For example the forward and backward move sets were updated incrementally, and the game position information was encoded in space-efficient manner.

The tests were run on an AMD Athlon64 3200+ computer with 1 GB RAM, Mandriva Linux 2006 OS and GCC 4.0.3 compiler.

3.1 Testing random sampling

We tested both RS1 and RS2 with sampling sizes $s = 1, 10, 100, 1000$ and 10000 to see how their performance scales. The test was repeated 20 times for each value of s . Table 1 lists the average numbers of moves found in these experiments. The corresponding average running times in seconds are shown inside parentheses.

SA

```

1  select a cooling ratio  $\alpha$  and an initial temperature  $t_0$ ;
   select a frozen temperature  $t_f$  and an equilibrium detection rate  $r$ ;
   set  $P$  to be the initial position of the game;
   set  $moves \leftarrow 0$ ; set  $c \leftarrow 0$ 
   initialize the forward and backward move sets  $F$  and  $B$ ;
2  while  $t \geq t_f$ 
3      do while  $c \leq r$ 
4          do  $c \leftarrow c + 1$ ;
5              randomly select a move  $m$  from  $F \cup B$ ;
6              if  $m \in F$ 
7                  then  $moves \leftarrow moves + 1$  and  $P \leftarrow P \cup m$ ;
8              else generate a random real  $q$ ,  $0 \leq q \leq 1$ ;
9                  if  $q \leq e^{-1/t}$  or  $F = \emptyset$ 
10                     then set  $moves \leftarrow moves - 1$  and  $P \leftarrow P \setminus m$ ;
11                     else select random  $m' \in F$ ;
12                         set  $moves \leftarrow moves + 1$  and  $P \leftarrow P \cup m'$ ;
13                 update the sets  $F$  and  $B$ ;
14                  $t = \alpha t$ ;
15                  $c = 0$ ;
16 return the best found solution over whole optimization process;
```

Algorithm 3.1: A simulated annealing (SA) algorithm for the Morpion Solitaire.

As expected, RS1 is much slower than RS2 with the same value of k as its sampling effort is multiplied by the average number of available moves. In practically all of the cases, the running time for RS1 is around 13 - 15 times higher than for RS2. It is also clearly evident how the average solution scores improve as k (and running time) is increased. This shows also in that with the same value of k , the more effort-making RS1 finds better solutions.

The two methods can be compared more fairly by looking at $k = 10^i$ with RS1 and $k = 10^{i+1}$ with RS2, as the running times are then fairly similar. There are no significant differences.

The best values found with RS1 and RS2 are listed later in Table 3. Both RS1 and RS2 were able to improve upon the previous results given in [4] for the models $G_3(A_3)$ and $G'_3(A_3)$, but they were not really good with $G_4(A_4)$ and $G'_4(A_4)$. Overall, it will be seen that simulated annealing turned out to be a significantly more competitive method.

Table 1: Statistics for RS1 and RS2.

Model (method)	Rep.	Average solutions (average running time in seconds)				
		$s = 10^0$	$s = 10^1$	$s = 10^2$	$s = 10^3$	$s = 10^4$
$G_3(A_3)$ (RS1)	20	30.6 (0.00)	31.9 (0.05)	32.9 (0.63)	33.3 (6.55)	33.8 (64.9)
$G_3(A_3)$ (RS2)	20	27.5 (0.00)	30.5 (0.00)	31.3 (0.04)	32.2 (0.46)	33.6 (4.69)
$G'_3(A_3)$ (RS1)	20	51.6 (0.01)	55.1 (0.15)	57.4 (1.88)	59.4 (20.3)	59.7 (209.0)
$G'_3(A_3)$ (RS2)	20	45.6 (0.00)	51.1 (0.01)	53.9 (0.13)	57.0 (1.46)	59.1 (15.8)
$G_4(A_4)$ (RS1)	20	60.1 (0.02)	61.5 (0.25)	63.0 (2.77)	63.9 (27.6)	64.4 (275.4)
$G_4(A_4)$ (RS2)	20	56.8 (0.00)	60.0 (0.01)	61.8 (0.19)	62.7 (1.94)	64.3 (20.2)
$G'_4(A_4)$ (RS1)	20	80.9 (0.03)	85.7 (0.42)	88.9 (4.85)	93.7 (52.3)	96.5 (556.9)
$G'_4(A_4)$ (RS2)	20	73.4 (0.00)	80.4 (0.03)	86.2 (0.35)	90.9 (3.60)	94.0 (37.3)

3.2 Testing simulated annealing

To detect a suitable temperature interval for SA, we performed the following tests with all four models. We tried initial temperatures $t_0 = 10, 9.8, 9.6, \dots, 0.4, 0.2$ with the frozen temperature $t_1 = t_0$, ie. equal to the initial temperature. This was intended to give clues about search performance with a certain temperature t . For the iteration count r , we tested the values $10^4, 10^5, 10^6$, and 10^7 . The experiment was repeated 20 times for each pairing of $t_0 = t_i$ and r values. The average results of the experiments are illustrated in Figure 4 for models $G_3(A_3)$ and $G'_3(A_3)$ and in Figure 5 for models $G_4(A_4)$ and $G'_4(A_4)$.

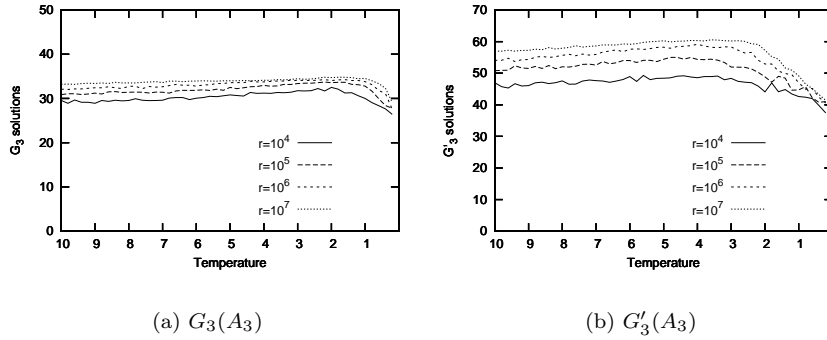


Figure 4: Average results for $G_3(A_3)$ and $G'_3(A_3)$ with different initial temperatures.

It is easy to see that there are no big differences in the average solution values of different temperatures under the disjoint model (see Figs. 4 (a) and 5 (a)). The average solutions became noticeably worse only when the temperature

decreased under 1.0. For the touching models, the best average solutions were obtained between temperatures 5.0 and 3.0 (see Fig. 4 (b)) and 10.0 and 6.0 (Fig. 4 (b)). This shows that SA achieves best solutions by performing almost completely random walks in the search space (between temperature interval 10.0...6.0 the probability of accepting backward moves lies between 0.90 and 0.84).

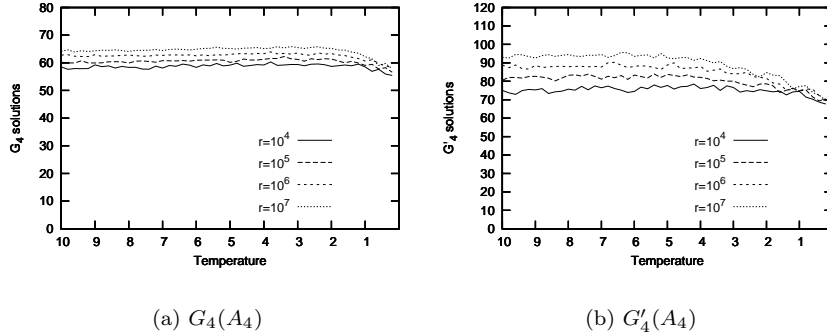


Figure 5: Average results for $G_4(A_4)$ and $G'_4(A_4)$ with different initial temperatures.

Table 2 reports average results and running times for temperatures that gave the highest average score in the parameter detection experiment. We have listed the number of repeats (Rep.), used temperature (Temp.), and the number of iterations for all models. The average running times in seconds are again shown inside parentheses.

Table 2: SA statistics.

Model	Rep.	Temp.	Average solutions (Average running time in seconds)			
			$r = 10^4$	$r = 10^5$	$r = 10^6$	$r = 10^7$
$G_3(A_3)$	20	1.8	32.1 (0.01)	33.7 (0.12)	34.2 (1.18)	34.8 (11.8)
$G'_3(A_3)$	20	3.6	48.9 (0.01)	54.5 (0.13)	58.2 (1.33)	60.6 (13.2)
$G_4(A_4)$	20	3.2	59.4 (0.01)	61.5 (0.14)	63.2 (1.38)	65.9 (13.7)
$G'_4(A_4)$	20	8.0	75.8 (0.02)	83.3 (0.15)	87.9 (1.53)	94.5 (15.4)

From the average results illustrated in Figures 4 and 5, and from the sample solutions given in Table 2, it can be seen that increasing the number of iterations (and thus the running time) results also in improving the average solutions.

After our preliminary experiments, we performed some further SA-runs with parameters allowing longer running time. With parameters $t_0 = 2.0$, $t_1 = 1.8$, $\alpha = 0.99999$, and $r = 10^6$, SA found a solution 74 for the $G_4(A_4)$ model. The running time was roughly 2 hours and 5 minutes. A solution 79 was found after several days computation by allowing even longer running time by increasing α and r , but due to a computer crash exact parameters, used seed and the running time were lost. A solution 143 was found for the $G'_4(A_4)$ model in roughly two days with parameters $t_0 = 10$, $t_1 = 9$, $\alpha = 0.9999$ and $r = 8 \times 10^7$. The solutions for models $G_3(A_3)$ and $G'_3(A_3)$ did not improve in these further runs.

Table 3 lists the overall best solution values we found with each method. In all cases SA has been able to find a better solution than either RS1 or RS2. A

comparison between Tables 1 and 2 does not reveal large differences between the average solutions produced by the three methods when they use roughly similar amounts of time. But nevertheless we could not find as good best results with RS1 and RS2 as with SA. One factor here is the fact that SA spends time in doing backward-moves, and SA also does not automatically play the game to the end. RS1 and RS2 always use complete endgames in sampling the search space.

Table 3: The best solutions.

	Model			
	$G_3(A_3)$	$G'_3(A_3)$	$G_4(A_4)$	$G'_4(A_4)$
The best RS1 solutions	34	61	66	105
The best RS2 solutions	34	61	66	100
The best SA solutions	35	62	79	143
Previous best	31	56	78	170

3.3 Further remarks

Our SA-based method was able to achieve a new record for three of the four interesting models with Malta Cross -shaped initial configuration. The record games are illustrated in Figs. 2 and 3.

In the remaining case, the model $G'_4(A_4)$, we managed to find a score 143, which seems rather low in comparison to the best known record of 170 moves. This record has been constructed by hand. The best known computer-generated results mentioned in literature are 122 by Juillé [9], 136 by Helmstetter and Cazenave [6], and reportedly Zimmer has achieved the score 143 by developing Juillé’s method further [1]. As our SA method is still in an early and immature development stage, we find the result satisfactory since it is equal to the current best computational record.

It was verified experimentally in [6] that the last 100 moves of the record 170 for $G'_4(A_4)$ are optimal. They also verified a similar result for their own computer-generated record of 136. This seems to indicate that the model $G'_4(A_4)$ is very challenging in that the chance of reaching a good score may be decided already in a surprisingly early stage of the game. This makes probabilistic search very difficult, as such a long systematic chain of moves has a very low probability of being tried. At the moment a human player has an advantage over computing by being able to see some repeating structure in the early stages of the game and being able to exploit that systematically.

We tested the above-discussed properties of $G'_4(A_4)$ by running SA from an initial setting that has been constructed by playing first c moves from the record game of 170 moves. When c was around 90 or higher, SA was consistently able to find the optimal result 170. But this ability was lost already around $c = 85$ or lower.

4 Conclusion

In this paper we presented random sampling and simulated annealing algorithms for Morpion Solitaire. Our experiments found that especially the SA-based method can be used to obtain good quality solutions for the game.

We improved three of the four non-trivial lower bounds by achieving new game records $G_3(A_3) \geq 35$, $G'_3(A_3) \geq 62$, and $G_4(A_4) \geq 79$.

Currently all backward and forward moves of the algorithms are selected purely by random selection (or sampling based on purely random endgames). This raises a question whether we could develop some smarter scoring/heuristic schemes for selecting move-selection.

References

- [1] <http://euler.free.fr/morpion.htm>.
- [2] Boyzy, B.: Associating domain-dependent knowledge and Monte Carlo approaches within a Go program. *Information Sciences* **175** (2005) 247–257.
- [3] Cazenave, T.: Reflexive Monte-Carlo Search, In *Proceedings of Computer Games Workshop*, pages 165–173, 2007.
- [4] Demaine, E.D., Demaine, M.L., Langerman, A., and Langerman, S.: Morpion Solitaire. *Theory of Computing Systems* **39** (2006) 439–453.
- [5] Flammenkamp, A.: Le morpion solitaire. Manuscript, March 19 2003. <http://wwwhomes.uni-bielefeld.de/achim/morpion.dvi>
- [6] Helmstetter, B. and Cazenave, T.: Incremental Transpositions. In *Proceedings of the 4th International Conference of Computers and Games*, volume 3846 of LNCS, pages 220–231, 2004.
- [7] Joris, W.: All Squared Up: Tic-Tac-Toe for Geniuses (100 Strategic Games for Pen and Paper). Carlton Books, London, 2002.
- [8] Juillé, H.: Incremental co-evolution or organism: a new approach for optimization and discovery of strategies. In *Proceedings of the Third European Conference on Artificial Life*, volume 929 of LNCS, pages 246–260, 1995.
- [9] Juillé, H.: Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm. PhD thesis, Department of Computer Science, Brandeis University (1999) 82–89.
- [10] Kirkpatrick, S., Gelatt, C., and Vecchi, M.: Optimization by simulated annealing, *Science* **220** (1983) 671–80.
- [11] Stefan Langerman. A game. <http://slef.org/jeu>.
- [12] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E.: Equation of state calculation by fast computing machines, *Journal of Chemical Physics* **21** (1953) 1087–1091.