Genet Program Evolvable Mach (2008) 9:281–294
DOI 10.1007/s10710-008-9062-1

ORIGINAL PAPER

Evolving strategy for a probabilistic game of imperfect information using genetic programming

Wojciech Jaśkowski · Krzysztof Krawiec · Bartosz Wieloch

Received: 31 December 2007 / Revised: 30 April 2008 / Published online: 10 July 2008

Abstract We provide the complete record of methodology that let us evolve *BrilliAnt*, the winner of the Ant Wars contest. Ant Wars contestants are virtual ants collecting food on a grid board in the presence of a competing ant. BrilliAnt has been evolved through a competitive one-population coevolution using genetic programming and fitnessless selection. In this paper, we detail the evolutionary setup that lead to BrilliAnt's emergence, assess its direct and indirect human-competitiveness, and describe the behavioral patterns observed in its strategy.

1 Introduction

The 2007 edition of Genetic and Evolutionary Computation Conference (GECCO, London, July 7–12, 2007) included the *Ant Wars* contest aimed at evolving a controller for a virtual ant that collects food in a square toroidal grid environment in the presence of a competing ant. In a sense, this game extends the Artificial Ant problem [14], a popular genetic programming benchmark, into the framework of a two-player game.

Ant Wars may be classified as a probabilistic two-person board game with imperfect information and partial observability. The game starts with 15 pieces of food randomly distributed over an 11×11 toroidal board and two players, called Ant 1 and Ant 2, placed at predetermined locations, (5, 2) for Ant 1 and (5, 8) for

W. Jaśkowski · K. Krawiec (✉) · B. Wieloch
Institute of Computing Science, Poznan University of Technology, Poznan, Poland
e-mail: kkrawiec@cs.put.poznan.pl

W. Jaśkowski
e-mail: wjaskowski@cs.put.poznan.pl

B. Wieloch
e-mail: bwieloch@cs.put.poznan.pl

Ant 2. No piece of food can be located in the ants' starting cells. An ant's field of view is limited to a square 5 $\times$ 5 neighborhood centered at its current location. An ant receives the complete information about the states (empty, food, enemy) of all cells within its field of view.

The game lasts for 35 turns per player. In each turn an ant moves into one of the eight neighboring cells. Ant 1 moves first. Moving into an empty cell has no effect. If an ant moves into a cell with food, it scores 1 point and the cell is emptied. Moving into the cell occupied by the opponent kills it: no points are scored, but only the survivor can go on collecting food until the end of the game. A game is won by the ant that reaches the higher score. In case of a tie, Ant 1 is the winner.

As the outcome of the game depends on spatial distribution of food pieces, the proper choice of the better of two players requires grouping multiple games into *matches* played on different boards. A match consist of 2 $\times$ k games played on k random boards generated independently for each match. To provide for fair play, the contestants play two games on the same board, in the first game taking roles of Ant 1 and Ant 2, and then exchanging these roles. We refer to such a pair of games as a *double-game*. To win a 2 $\times$ k-games match, an ant has to win $k + 1$ or more games. In the case of tie, the total score determines the match outcome. If there is still a tie, a randomly selected contestant wins.

The contest rules required an ant's controller to be encoded as an ANSI-C function *Move*(*grid, row, column*), where *grid* is a two-dimensional array representing the board state, and (*row, column*) represents the ant's position. The function indicates the ant's next move by returning the direction encoded as an integer from interval [0,7]. The source code of the function was not allowed to exceed 5 kB in length.

In this paper, we tell the story of *BrilliAnt*, the Ant Wars winner submitted by our team. BrilliAnt has been evolved through competitive one-population coevolution using genetic programming and a fitnessless selection method. We assess BrilliAnt's human-competitiveness in both direct terms (playing against a human opponent) and indirect terms (playing against a human-devised strategy), and analyze its behavioral patterns.

2 Evolving game strategies

Though games were at the center of AI's interest since its very beginning, achieving a human-competitive performance in this area without an intense help of human expertise and computational power is still beyond our reach. In the 1960s and 1970s game playing strategies, like the famous Bernstein's chess and Samuel's checker players, were hand-crafted by humans. The most spectacular achievement of artificial intelligence in the game domain was the grand master Garry Kasparov's defeat in a duel with Deep Blue. However, Deep Blue implements a brute force approach heavily based on human expertise so it does not contribute much to the understanding of human intelligence. In the domain of the Go game, the \$1.4 M Ing Prize for the first computer to beat a nominated human competitor has never been touched,[1] presumably because Go has too many states to be approached by brute force. Hard AI is also often helpless when it comes to the real-time (strategy) games [3] or multi-agent games where the number of possible states can be even greater than in Go. Also, things get more complicated also for the hand-designed algorithms when the game state is only partially-observable or the game is probabilistic by nature.

This experience clearly indicates that handcrafting a good game-playing strategy for a nontrivial game is a serious challenge. The hope for progress in the field are the methods that automatically construct the game playing programs, like the coevolution used in our approach.

Coevolution was successfully applied to many two-person games, including Backgammon [21], Checkers [9], NERO [26], Blackjack [4], Pong [19] and a small version of Go [15]. Coevolution itself has recently been heavily studied [6–8] to guarantee monotonic progress towards the selected solution concept.

Within genetic programming (GP, [14]), Koza was the first who used it to evolve strategies [13] for a simple discrete two-person game. Since then, it has been demonstrated many times that the symbolic nature of GP is suitable for this kind of task. Past studies on the topic include both trivial games such as Tic Tac Toe [1] or Spoof [29], as well as more complicated and computationally-demanding games, like poker [25]. Core Wars, a game in which two or more programs compete for the control of the virtual computer, is among a popular benchmark problem for evolutionary computation and one of the best evolved players was created using GP [5]. Luke's work [16] on evolving soccer softball team for RoboCup97 competition belongs to the most ambitious applications of GP to game playing since it involved a complicated environment and teamwork. Recently, Sipper and his coworkers demonstrated [24] human-competitive GP-based solutions in three areas: backgammon [2], RoboCode [23] (tank-fight simulator) and chess endgames [10].

3 Strategy encoding

An ant's field of view (FOV) contains 25 cells and occupies 20.7% of board area, so, assuming an unbiased random distribution of food pieces, the expected number of visible food pieces is 3.02 when the game begins. The probability of having n food pieces within the FOV drops quickly as n increases; for instance, for $n = 8$ amounts to less than 0.5%. Thus, most of the food to be collected is usually beyond the FOV. Also, a reasonable strategy should obviously take into account the rotational invariance and symmetry of FOV; e.g., for two mirrored FOV states, an ant should behave in a mirrored way. These facts let us conclude that the number of distinct and likely FOV states is relatively low, and that a strategy based only on the *observed* FOV state cannot be competitive in a long run. It seems reasonable to virtually extend the FOV by keeping track of the past board

[1] The Ing Foundation stopped to funding the prize in 2000.

states. Thus, we equip our ants with *memory*, implemented by three arrays that are overlaid over the board:

- *Food memory F*, which keeps track of food locations observed in the past,
- *Belief table B*, which describes ant's belief in the current board state,
- *Track table V*, which stores the cells already visited by ant.

After each move, we copy food locations from the ant's FOV into F. Within the FOV, old states of F are overridden by the new ones, while F cells outside the current FOV remain intact. As the board state may change subject to opponent's actions and make the memory state obsolete, we also simulate a memory decay in the belief table B. Initially, the belief for all cells is set to 0. Belief for the cells within the FOV is always 1, while outside the FOV it fades exponentially by 10% with each move. Table V, initially filled with zeros, stores ant's 'pheromone track' by setting the visited elements to 1.

To represent our ants we used the tree-based strongly typed genetic programming [20]. A GP tree is expected to evaluate the utility of the move in a particular direction: the more attractive the move, the greater the tree's output. To provide for rotational invariance, we evaluate multiple orientations using the same tree. However, as the ants are allowed to move both straight and diagonally, we store *two* trees in each individual, one for handling the straight directions (N, E, S, W) and one to handle the diagonal directions (NE, NW, SE, SW).[2] Given a particular FOV state, we present it to the trees by appropriately rotating the FOV, the remaining part of the board, and the memory, by a multiple of 90°, and querying both trees each time. Among the eight obtained values, the maximum response indicates the most desired direction and determines the ant's next move; ties are resolved by preferring the earlier maximum.

We define three data types: *float* (F), *boolean* (B), and *area* (A). The area type represents a rectangle stored as a quadruple of numbers: dimensions and midpoint coordinates (relative to ant's current position, modulo board dimensions). To avoid considering exceedingly large areas, we constrain the sum of area dimensions to six by an appropriate genotype-to-phenotype mapping. For instance, the dimensions encoded as (2, 5) in the genotype are effectively mapped to (1, 4) during tree execution.

The GP function set and the terminals are presented in Tables 1 and 2. Note that some functions calculate their return values not only from the actual state of the board, but also from the food memory table F and the belief table B. For example, NFood(A) returns the scalar product of table F (food pieces) and table B (belief), constrained to area A.

It is worth emphasizing that all GP functions used here are straightforward. Even the most complex of them boil down to counting matrix elements in designated rectangular areas. Though one could easily come up with more sophisticated functions, this would contradict the rules of the contest, which promoted the evolved rather than the designed intelligence.

[2] We considered using a single tree and mapping the diagonal board views into the straight ones; however, this leads to significant topological distortions which could deteriorate ant's perception.

Table 1 The terminals used by evolving strategies

Terminal	Interpretation
Const()	An ephemeral random constant (ERC) for type F ([−1; 1])
ConstInt()	An integer-valued ERC for type F (0..5)
Rect()	An ERC for type A
TimeLeft()	The number of moves remaining to the end of the game
Points()	The number of food pieces collected so far by the ant
PointsLeft()	Returns 15-Points()
FoodHope()	Returns the maximal number of food pieces that may be reached by the ant within two moves (assuming the first move is made straight ahead, and the next one in an arbitrary direction)

Table 2 The non-terminals

Non-terminal	Interpretation
IsFood(A)	Returns *true* iff A contains at least one piece of food
IsEnemy(A)	Returns *true* iff A contains the opponent
And(B, B)	Logic functions
Or(B, B)	
Not(B)	
IsSmaller(F, F)	Arithmetic comparators
IsEqual(F, F)	
Add(F, F)	Scalar arithmetics
Sub(F, F)	
Mul(F, F)	
If(B, F, F)	The conditional statement
NFood(A)	The number of food pieces in the area A
NEmpty(A)	The number of empty cells in the area A
NVisited(A)	The number of cells already visited in the area A

4 The experiment

To limit human intervention, our ants undergo competitive evaluation, i.e., face each other, rather than an external selection pressure. In such a setup, termed one-population coevolution [18] or competitive fitness environment [16, 1], the fitness of an individual depends on the results of the games played with other individuals from the same population. The most obvious variant of this approach is the *round-robin tournament*, which needs $n(n-1)/2$ games to be played in each generation, where n is the population size. Another, computationally less demanding method is the *single-elimination tournament* proposed by Angeline and Pollack [1] (requires only $n-1$ games) or *k-random opponents* [22] (kn games). These fitness assignment methods were designed to be compatible with the evaluation-selection-recombination mantra characteristic for the traditional fitness-based evolutionary computation. Games played in the evaluation phase determine the individual's fitness that is subsequently used in the selection phase. After second thought, this scheme turns out to be redundant. Playing games is selective by nature, so why not use them directly for selection?

This observation led us to propose the approach of *fitnessless coevolution*. The key idea is to combine the typical evaluation and selection phases of evolutionary algorithm into one step of *fitnessless selection* [12]. Technically, we skip the evaluation and proceed directly to selection, which works like a single-elimination tournament played between k individuals randomly drawn from the population. The winner of this tournament becomes the result of the selection. The only factor that determines the winner is the specific sequence of wins and losses, so that no explicit fitness measure is involved in this process. This makes our approach significantly different from most of the methods presented in literature. The only related contribution known to us is [27], where Tettamanzi describes *competitive selection*—a form of stochastic binary tournament selection. For $k = 2$, our fitnessless selection is identical to competitive selection.

To make the initial decisions about the experimental setup and parameter settings, we ran some preliminary experiments. Finally we decided to set the run length to around 1,500 generations and, to effectively utilize the two-core processor, to employ the island model [28] with two populations, each of approximately 2,000 individuals. In all experiments, we used probabilities of crossover, mutation, and ERC mutation, equal to 0.8, 0.1, and 0.1, respectively. GP trees were initialized using ramped half-and-half method and were not allowed to exceed the depth of 8. The experiment was implemented in the ECJ [17] framework and for the remaining parameters we used the ECJ's defaults.

We relied on the default implementation of GP mutation and crossover available in ECJ, while providing specialized ERC mutation operators for particular ERC nodes. For Const() we perturb the ERC by a random, normally distributed value with mean 0.0 and standard deviation 1/3. For ConstInt(), we perturb the ERC by a random, uniformly distributed integer value from interval $[-1; 1]$. For Rect(), we perturb each rectangle coordinate or dimension by a random, uniformly distributed integer value from interval $[-1; 1]$. In all cases, we trim the resulting values to domain intervals.

To speed up the selection process and to meet the contest rules that required the ant code to be provided in C programming language (ECJ is written in Java), in each generation we serialize the entire population into one large text file, encoding each individual as a separate C function. The resulting file is then compiled and linked with the game engine, also written in C. The resulting executable is subsequently launched and carries out the selection, returning the identifiers of the selected individuals to ECJ. The compilation overhead is reasonably small, and it is paid off by the speedup provided by using C language. This approach allows us also to monitor the actual size of C code, constrained by the contest rules to 5kB per individual.

The best ant emerged in an experiment with 2,250 individuals evolving for 1,350 generations, using the fitnessless selection with tournament size $k = 5$ (thus 4 matches per single-elimination tournament), and with 2×6 games played in each match. We named it *BrilliAnt*, submitted to the Ant Wars competition, and won it. BrilliAnt not only evolved, but was also selected in a completely autonomous way, by running a round-robin tournament between all 2,250 individuals from the last generation of the evolutionary run. This process was computationally demanding: having only one double-game per match, the total number of games needed was more than 5,000,000, an equivalent of about 47 generations of evolution.

We assessed BrilliAnt's human-competitiveness with respect to two variants of this notion: *direct competitiveness*, i.e., the performance of the evolved solution playing against a human, and *indirect competitiveness*, meant as the performance of the evolved solution playing against a *program* designed by a human. For the former purpose, we implemented a software simulator that allows humans to play games against an evolved ant. Using this tool, an experienced human player played 150 games against BrilliAnt, winning only 64 (43%) of them and losing the remaining 86 games (57%). We developed also an online version of this tool that allows everybody play with BrilliAnt. At the time of writing, the collected statistics confirm the above result: 335 games won by BrilliAnt vs. 188 won by humans, and one draw. Even if we assume that inexperienced beginners account for great part of this statistics, these figures clearly indicate that the strategy elaborated by our approach is challenging for humans. The reader is encouraged to visit the Web page [11] and measure swords with BrilliAnt.

To analyze BrilliAnt's indirect competitiveness, we let it play against human-designed strategies—*humants*. We manually implemented several humants of increasing sophistication (*SmartHumant, SuperHumant*, and *HyperHumant*). HyperHumant, the best humant we could develop, memorizes the states of board cells observed in the past, plans 5 moves ahead, uses a probabilistic memory model, and implements several end-game rules (e.g., *when your score is 7, eat the food piece even if the opponent is next to it*).

Table 3 and Fig. 1 present the results of the round-robin tournament between the three humants, BrilliAnt, and four other evolved ants (*ExpertAnt, EvolAnt1, EvolAnt2, EvolAnt3*). Each pair of strategies played a match of 100,000 double-games. An arrow leading from *a* to *b* means that *a* turned out to be statistically better than *b*; no arrow means no statistical advantage (at 0.01 level[3]). Note that one of the evolved individuals, *ExpertAnt*, won 50.12% of games against HyperHumant. As BrilliAnt turned out to be worse than HyperHumant (loosing 52.02% of games), ExpertAnt could be considered a better pick for the Ant Wars contest. However, ExpertAnt has been selected by explicitly testing all ants from the last generation against the *manually designed* HyperHumant. BrilliAnt, on the contrary, evolved and was selected completely autonomously, so it has been appointed as our contestant.

[3] We estimate the statistical significance of the outcome of a match from the tails of the binomial distribution assuming the probability of success of .5 (the zero hypothesis is that both players win the same number of games, i.e., 50,000 games in this context).

Table 3 The results of a round-robin tournament between the evolved ants (in bold) and humants (plain font). Maximum possible score is 21,000,000

Player	Matches won	Games won	Total score
ExpertAnt	6	760,669	10,598,317
HyperHumant	6	754,303	10,390,659
BrilliAnt	6	753,212	10,714,050
EvolAnt3	3	736,862	10,621,773
SuperHumant	3	725,269	10,130,664
EvolAnt2	3	721,856	10,433,165
EvolAnt1	1	699,320	10,355,044
SmartHumant	0	448,509	9,198,296

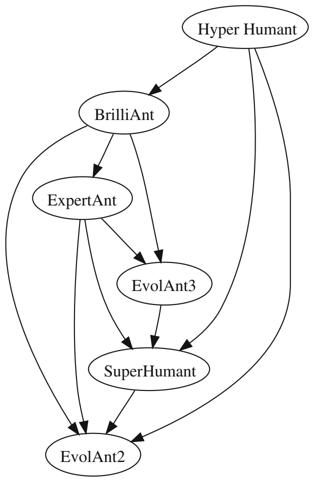


Fig. 1 Graph showing relations between players. If player *a* is statistically better than player *b* ($p = 0.01$), an arrow leads from *a* to *b* . If none of them is better, no arrow is drawn. EvolAnt1 and SmartHumant were not showed to improve graph's readability

5 BrilliAnt's strategy

BrilliAnt's genotype (a pair of GP trees) contains 237 nodes, so its presentation, not mentioning analysis, would be overly long and is beyond the scope of this paper. However, we are still able to analyze its phenotype, meant as its behavior in the course of game playing. BrilliAnt exhibits a surprisingly rich repertoire of behavioral patterns, ranging from obvious to quite sophisticated ones. Faced with the FOV with two corner areas occupied by food, BrilliAnt always selects the direction that gives chance for more food pieces. It reasonably handles the trade-off between food amount and food proximity, measured using the chessboard (Chebyshev) distance (the minimal number of moves required to reach a cell). For instance, given a group of two pieces of food at distance 2 ((2, 2) for short), and a group of two pieces of food at distance 1, i.e., (2, 1), BrilliAnt chooses the latter option, i.e., $(2,2) \prec (2,1)$ using a shorthand notation. Similarly, $(1,1) \prec (2,2), (3,2) \prec (2,1), (3,2) \prec (3,1)$, and

$(2,2) \prec (3,2)$. If both food groups contain the same number of food pieces but one of them is accompanied by the opponent, BrilliAnt chooses the other group.

Food pieces sometimes happen to arrange into ‘trails’, similar to those found in the Artificial Ant benchmarks [14]. BrilliAnt perfectly follows such paths as long as the gaps between trail fragments are no longer than two cells (see Fig. 2a for an example). However, when faced with a large isolated group of food pieces, it does not always consume them in an optimal order, i.e., in a minimum number of moves.

If the FOV does not contain any food, BrilliAnt proceeds in the NW direction. However, as the board is toroidal, keeping moving in the same direction makes sense only to a certain point, because it brings the player back to the starting point after 11 moves, with a significant part of the board still unexplored. Apparently, evolution discovered this fact: after seven steps in the NW direction (i.e., when FOV starts to intersect with the initial FOV), BrilliAnt changes its direction to SW, pursuing the following sequence: 7NW, 1SW, 1NW, 1SW, 6NW, 1SW, 1NW. A simple analysis reveals that these 18 moves, shown in Fig. 2b, provide the complete coverage of the board. This behavior seems quite efficient, as the minimal number of moves that scan the entire board is 15. Note also that this contains only diagonal moves. In absence of any other incentives, this is a locally optimal choice, as a diagonal move uncovers nine board cells, while a non-diagonal one uncovers only five of them.

Evolving this full-board scan is quite an achievement, as it manifests in absence of food, a situation that is close to impossible in Ant Wars, except for the highly unlikely scenario of the opponent consuming all the food earlier. BrilliAnt exhibits variants of this behavioral pattern also after some food pieces have been eaten and its FOV is empty.

BrilliAnt also makes reasonable use of its memory. When confronted with multiple groups of food pieces, it chooses one of them and, after consuming it,

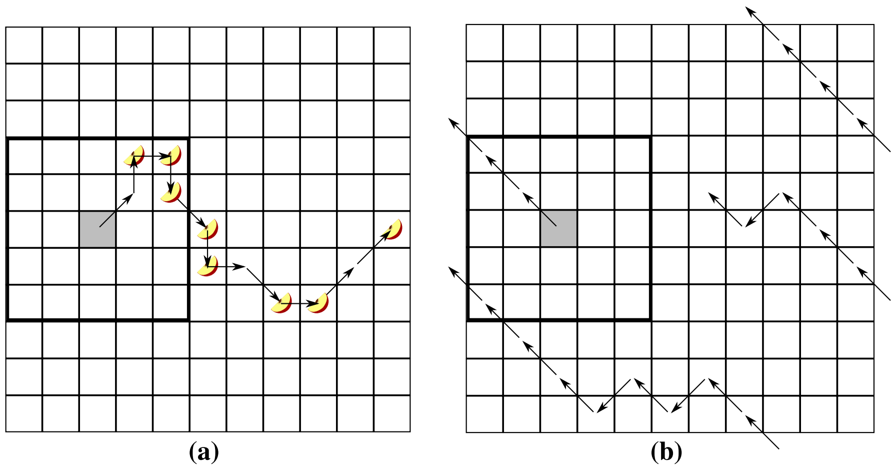


Fig. 2 BrilliAnt’s behaviors when following a trail of food pieces (**a**), and in absence of food (**b**). Gray cell and large rectangle mark BrilliAnt’s starting position and initial FOV, respectively

returns to the other group(s), unless it has spotted some other food in the meantime. This behavior is demonstrated in Fig. 3a where BrilliAnt, faced with the initial board state with four food pieces visible in the corners of FOV, follows an almost optimal trajectory. Note that as soon as it makes the first NW move, three food pieces disappear from the FOV, so memory is indispensable here. After completing this task, BrilliAnt proceeds to the unexplored parts of the board.

BrilliAnt usually avoids the opponent, unless it comes together with food and no other food pieces are in view. In such a case, it approaches the food, maintaining at least distance 2 from the opponent. For an isolated food piece, this often ends in a deadlock: the players hesitatingly walk in the direct neighborhood of the food piece, keeping safe distance from each other. None of them can eat the piece, as the opponent immediately kills such a daredevil. This behavior is shown in Fig. 3b, where BrilliAnt is the ant with starting position marked by the gray cell, and the numbers reflect ants' positions in consecutive turns (BrilliAnt moves first). After making the first move, BrilliAnt spots the opponent on the other 'side' of food, so it does not eat the piece but walks along it (moves 2 and 3). In the meantime, the opponent behaves analogously. In absence of any other incentives, this behavior could last till the end of game. However, as soon as BrilliAnt spots another piece of food (after making move #3 in Fig. 3b), it changes its mind and starts heading towards it, leaving the disputed food piece to the opponent.

BrilliAnt also learned how to resolve such deadlocks. When the end of game comes close and the likelihood of finding more food becomes low, it may pay off to sacrifice one's life in exchange for food—there will be not much time left for the opponent to gather more food. This in particular applies to the scenario when both players scored 7 and the food piece of argument is the only one left. This 'kamikaze' behavior emerged also in other evolutionary runs. Figure 4b illustrates this behavior in terms of the death rate statistic for one of the experiments. The ants from the several initial generations play poorly and are likely to be killed by the opponent. With time, they learn how to avoid the enemy and, usually at 200–300th generation,

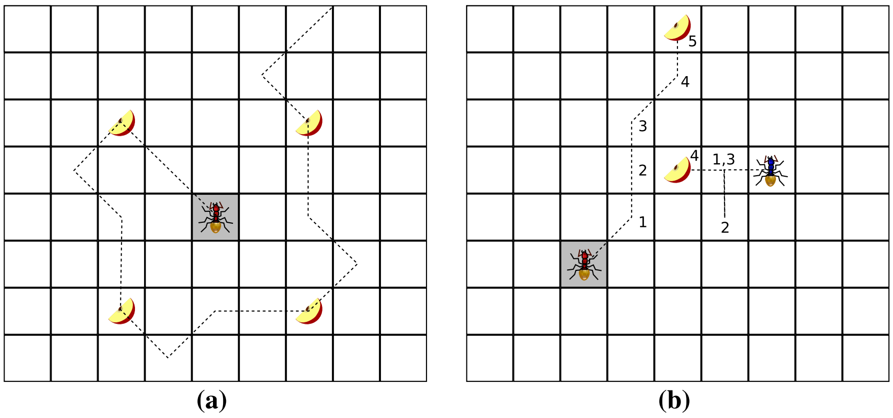
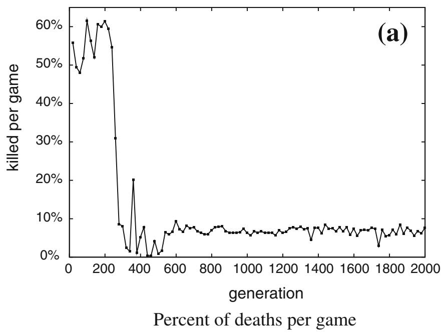


Fig. 3 (**a**) BrilliAnt's memory at work. (**b**) The deadlock situation and its resolution



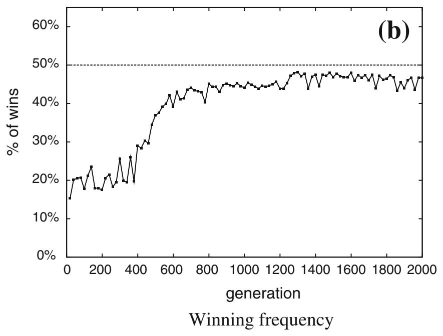


Fig. 4 The dynamics of a typical evolutionary run. Each point corresponds to the best-of-generation ant chosen on the basis of 2 $\times$ 250 games against HyperHumant

the best ants become perfect at escaping that threat (see Fig. 4b). Then, around 400–500th generation, the ants discover the benefit of the 'kamikaze' strategy, which results in a notable increase of death rate, but pays off in terms of the winning frequency.

BrilliAnt is also able to correctly estimate its chances of reaching a piece of food before the (visible) opponent, while taking into account the risk of being eaten (see Fig. 5; in all scenarios shown here BrilliAnt, located at the center, moves first). In Fig. 5a, BrilliAnt decides to approach the food because its first move effectively repels the opponent. In Fig. 5b, on the contrary, BrilliAnt walks away as it has no chance of reaching the food piece before the opponent (the shortest path traverses the cell controlled by the opponent). The situation depicted in Fig. 5c gives equal chances to both players to reach the food piece, so BrilliAnt approaches it, maintaining a safe distance from the opponent. If the opponent moves analogously to the north, this may end up in a deadlock described earlier. However, on the way to the food piece BrilliAnt or the opponent may spot another food piece and walk away, so this behavior seems reasonable. If there is a choice between an uncertain food piece and food piece that may be reached at full safety, BrilliAnt chooses the latter option, as shown in Fig. 5d.

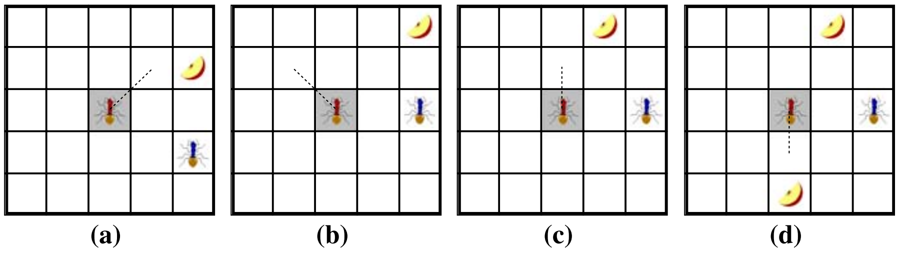


Fig. 5 BrilliAnt's behavior when faced with food and an opponent

6 Conclusions

We described evolution of game strategies that uses relatively little domain knowledge, where both the evolution and the selection of the best-of-run individual are completely autonomous and do not involve any external (e.g., human-made) strategies. The evolved players are human-competitive in both direct and indirect sense and make reasonable choices based on the visible part of the board as well as on the memory state. In particular, BrilliAnt's traits include close-to-optimal board scanning in search of food, collecting of previously seen and memorized food pieces, the ability to cancel the previous plans after discovering new food pieces, and rational estimation of chances of reaching the food before the opponent. Also, BrilliAnt's strategy is dynamic, i.e., changing with the stage of the game, as demonstrated by its ability of sacrifice in exchange of food.

The mechanism of fitnessless selection lets the individuals play games against each other and propagates the winner to the next generation, making the existence of the objective fitness unnecessary. Though unusual from the viewpoint of the core EC research, selection without fitness has some rationale. The traditional fitness function is essentially a mere technical means to impose the selective pressure on the evolving population. It is often the case that, for a particular problem, the definition of fitness does not strictly conform its biological counterpart, i.e., the a posteriori probability of the genotype survival. By eliminating the need for an arbitrary numeric fitness, we avoid the subjectivity that its definition is prone to.

This approach may be directly applied to any two-player game, and seems to be especially appealing for game-related applications of evolutionary computation. In [12] we provide an in-depth analysis of fitnessless selection and show that it is dynamically equivalent to regular fitness-based evolution provided the fitness function fulfills a simple condition.

We find this results encouraging as the Ant Wars game is not trivial, mainly due to the partial observability of board state and imperfect information about opponent's behavior. There is no better way to appreciate the effectiveness of the simulated evolution than through a personal experience. Check at [11] if you can beat BrilliAnt.

Acknowledgments The authors wish to thank the anonymous reviewers for valuable feedback and discussion on this work. This research has been supported by the Ministry of Science and Higher Education grant # N N519 3505 33.

References

1. P.J. Angeline, J.B. Pollack, Competitive environments evolve better solutions for complex tasks, in *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, University of Illinois at Urbana-Champaign, 17–21 July 1993, ed. by S. Forrest (Morgan Kaufmann, 1993), pp. 264–270
2. Y. Azaria, M. Sipper, GP-gammon: genetically programming backgammon players. Genet. Prog. Evol. Mach. **6**(3), 283–300 (2005)
3. M. Buro, Real-time strategy games: a new AI research challenge, in *IJCIA*, ed. by G. Gottlob, T. Walsh (Morgan Kaufmann, San Francisco, 2003), pp. 1534–1535

4. J.B. Caverlee, A genetic algorithm approach to discovering an optimal blackjack strategy, in *Genetic Algorithms and Genetic Programming at Stanford 2000*, Stanford Bookstore, Stanford, CA, 94305-3079 USA, June 2000, ed. by J.R. Koza, pp. 70–79
5. F. Corno, E. Sanchez, G. Squillero, On the evolution of corewar warriors, in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Portland, OR, 20–23 June 2004 (IEEE Press, 2004), pp. 133–138
6. E. de Jong, The maxsolve algorithm for coevolution, in *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, Washington DC, USA, 25–29 June 2005, ed. by H.-G. Beyer, U.-M. O'Reilly, D.V. Arnold, W. Banzhaf, C. Blum, E.W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J.A. Foster, E.D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G.R. Raidl, T. Soule, A.M. Tyrrell, J.-P. Watson, E. Zitzler, vol. 1 (ACM Press, 2005), pp. 483–489
7. E.D. de Jong, A monotonic archive for pareto-coevolution. Evol. Comput. **15**(1), 61–93 (2007)
8. S. Ficici, J. Pollack, A game-theoretic memory mechanism for coevolution, in *Genetic and Evolutionary Computation*, Chicago, July 2003, ed. by E. Cantú-Paz, J. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. Potter, A.C. Schultz, K. Dowsland, N. Jonoska, J. Miller. Lecture Notes in Computer Science, vol. 2723 (Springer, 2003), pp. 286–297
9. D.B. Fogel, *Blondie24: Playing at the Edge of AI* (Morgan Kaufmann Publishers Inc., San Francisco, CA, 2002)
10. A. Hauptman, M. Sipper, Evolution of an efficient search algorithm for the mate-in-N problem in chess, in *Proceedings of the 10th European Conference on Genetic Programming*, vol. 4445 of *Lecture Notes in Computer Science*, Valencia, Spain, 11–13 Apr. 2007, ed. by M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, A.I. Esparcia-Alcázar (Springer, 2007), pp. 78–89
11. W. Jaśkowski, K. Krawiec, B. Wieloch, AntWars Applet, 2007, http://www.cs.put.poznan.pl/kkrawiec/antwars/
12. W. Jaśkowski, K. Krawiec, B. Wieloch, Fitnessless coevolution, in *GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, 2008
13. J.R. Koza, Genetic evolution and co-evolution of game strategies, in *The International Conference on Game Theory and Its Applications*, Stony Brook, New York, 15 July, 1992
14. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA, USA, 1992)
15. A. Lubberts, R. Miikkulainen, Co-evolving a go-playing neural network, in *Coevolution: Turning Adaptive Algorithms upon Themselves*, San Francisco, CA, USA, 7 July 2001, ed. by R.K. Belew, H. Juillè, pp. 14–19
16. S. Luke, Genetic programming produced competitive soccer softbot teams for robocup97, in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, WI, USA, 22–25 July 1998, ed. by J.R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba, R. Riolo (Morgan Kaufmann, 1998), pp. 214–222
17. S. Luke, ECJ Evolutionary Computation System, 2002, http://cs.gmu.edu/eclab/projects/ecj/
18. S. Luke, R.P. Wiegand, When coevolutionary algorithms exhibit evolutionary dynamics, in *Workshop on Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms (at GECCO 2002)*, ed. by A. Barry (AAAI Press, New York, 2002), pp. 236–241
19. G.A. Monroy, K.O. Stanley, R. Miikkulainen. Coevolution of neural networks using a layered pareto archive, in *GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, USA, 8–12 July 2006, ed. by M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M.V. Butz, C. Coello Coello, D. Dasgupta, S.G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, D. Thierens, vol. 1 (ACM Press, 2006), pp. 329–336
20. D.J. Montana, Strongly typed genetic programming. Evol. Comput. **3**(2), 199–230 (1995)
21. J.B. Pollack, A.D. Blair, Co-evolution in the successful learning of backgammon strategy. Mach. Learn. **32**(3), 225–240 (1998)
22. C. Reynolds, Competition, coevolution and the game of tag, in *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by R.A. Brooks, P. Maes (MIT Press, 1994), pp. 59–69
23. Y. Shichel, E. Ziserman, M. Sipper, GP-robocode: using genetic programming to evolve robocode players, in *Proceedings of the 8th European Conference on Genetic Programming*, Lausanne, Switzerland, 30 Mar.–1 Apr. 2005, ed. by M. Keijzer, A. Tettamanzi, P. Collet, J.I. van Hemert, M. Tomassini. Lecture Notes in Computer Science, vol. 3447 (Springer, 2005), pp. 143–154

24. M. Sipper, Attaining human-competitive game playing with genetic programming, in *Proceedings of the 7th International Conference on Cellular Automata, for Research and Industry, ACRI*, Perpignan, France, Sept. 20–23 2006, ed. by S.E. Yacoubi, B. Chopard, S. Bandini. Lecture Notes in Computer Science, vol. 4173 (Springer, Invited Lectures), p. 13
25. K.C. Smilak, Finding the ultimate video poker player using genetic programming, in *Genetic Algorithms and Genetic Programming at Stanford 1999*, Stanford Bookstore, Stanford, CA, 94305-3079 USA, 15 Mar. 1999, ed. by J.R. Koza, pp. 209–217
26. K. Stanley, B. Bryant, R. Miikkulainen, Real-time neuroevolution in the NERO video game. IEEE Trans. Evolut. Comput. **9**(6), 653–668 (2005)
27. A.G.B. Tettamanzi, Genetic programming without fitness, in *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28–31, 1996*, Stanford University, CA, USA, 28–31 July 1996, ed. by J.R. Koza (Stanford Bookstore, 1996), pp. 193–195
28. D. Whitley, S. Rana, R. Heckendorn, The island model genetic algorithm: on separability, population size and convergence. J. Comput. Inform. Technol. **7**(1), 33–47 (1999)
29. M. Wittkamp, L. Barone, Evolving adaptive play for the game of spoof using genetic programming, in *Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games (CIG06)*, University of Nevada, Reno, campus in Reno/Lake Tahoe, USA, 22–24 May 2006, ed. by S.J. Louis, G. Kendall (IEEE Press, 2006), pp. 164–172