EVOLVING STRATEGY FOR GAME PLAYING

Josef Hynek
University of Hradec Králové
Faculty of Informatics and Management
Department of Quantitative Methods and Informatics
Nejedlého 573, 500 03 Hradec Králové
Czech Republic
Josef.Hynek@uhk.cz

ABSTRACT

This paper examines genetic algorithm and machine learning using the game of Nim. We have studied various attempts to evolve a competitive or even optimal strategy for this game that have been undertaken before. Based on these findings we have reviewed them and then we have designed a new approach that has been tested on a particular version of the game of Nim. Contrary to the evolving populations of "hosts" and "parasites", we have proposed a solution that is based on a genetic algorithm utilizing single population only. Moreover, we have exploited a kind of macromutation operator previously utilized within the field of genetic programming. The so called headless chicken crossover helped us to significantly speed up the evolutionary process. We have carried out series of experiments and the analysis of these experiments is presented here. We do believe that the approaches and results described here can be useful when tackling other problems where the suitable strategy goal is pursued.

INTRODUCTION

Game playing in general is very popular amongst the artificial intelligence community and many search methods have been studied and designed in this particular field. Numerous games have been thoroughly studied, tested and tackled by the variety of methods since the very beginning of this scientific field. These methods range from the simplest brutal force search strategies to miscellaneous sophisticated heuristic techniques.

Our aim here is to examine utilization of genetic algorithms to evolve game playing strategy. Genetic algorithms make use of a "survival of the fittest" rule that gives them enough power to eliminate poor strategies, make it more advantageous for better ones, and utilizing suitable genetic operators and basic principles of inheritance to create a new offspring. Repeating this "naturally inspired life cycle" many times we usually obtain a very good or even optimal strategy for a given task.

To demonstrate the power of genetic algorithms we have decided to employ the game of Nim. There are several different versions of this game and because of its simple laws the game is frequently used to present various techniques of artificial intelligence. Moreover, for certain instances of the game we are going to describe and to exploit it for testing purposes here, the optimal (winning) strategy is known. It gives us a unique opportunity to measure the outcomes of our algorithm and to compare them with the known optimal strategy.

There are many books and papers on game playing and genetic algorithms and that is why we will try to narrow our focus here on those directly related to our paper only (for a broader overview see for example [1]). When searching for the former attempts to utilize genetic algorithms to develop a strategy for the game of Nim, we have found especially two important papers of Rosin and Belew [8, 9]. They have considered such a version of the game of Nim where the initial configration consists of four piles containing 3, 4, 5, and 4 stones. Players alternate removing an arbitrary number of stones from a single pile and the player to take the last stones wins. This configuration allows the first player to force a win with optimal play [8]. They have described their experiments when exploring various sampling methods there. Their results are quite impressive and so it stimulated our interest to run our own experiments on a different version of the game of Nim. Moreover, we have utilized one population of individuals only instead of two distinctive populations of "hosts" and "parasites".

PROBLEM DESCRIPTION

There are several versions of the game of Nim. We have decided to experiment with the form where N (N>0) stones are placed on the table and two players alternate to remove m (m>0) stones in each step. There is a given number k>0 that restricts the maximal number of stones to be removed in one turn by each player and thus $0<m\leq k$. The player who takes the last stone from the table looses the game.

The good feature of this game (from the point of a researcher) is that there is a known wining strategy that depending on the initial number of stones on the table assures secure win to the relevant player.

The strategy can be easily derived from the definition of the game. To become a winner, our aim is to leave only one last stone for our opponent on the table at the final stage of the game. It is clear that our opponent can take at least 1 and at maximum *m* stones in one turn, which means that we can easily guarantee that *m+1* stones can be taken off the table in each run. Consequently, if there are *(m+1)*x+1* stones on the table where *x* is arbitrary positive natural number and our opponent is going to take his move, we can see that it is rather straightforward to achieve in *x* rounds the position that there will be the only one stone on the table left for our opponent. This strategy can be formulated in three rules:

1. If there is only one stone on the table you have to take it and it means that this game is lost.
2. If there are n (n>1) stones on the table it is an optimal decision to take y stones where

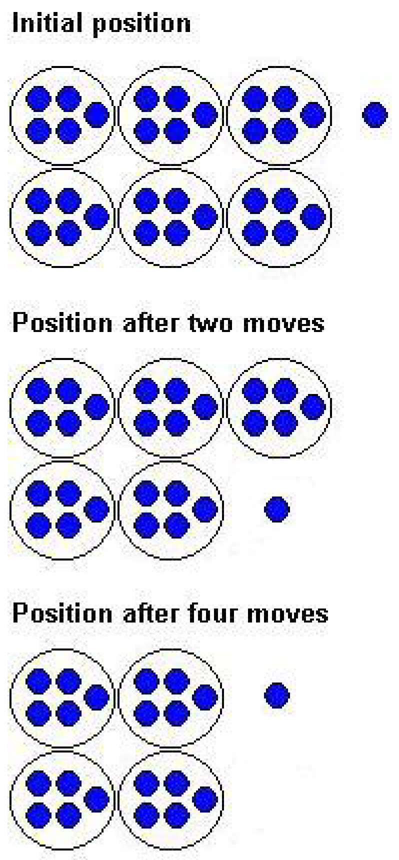
   $$y=(n-1) \bmod (k+1)$$

   providing that $y > 0$.
3. If y computed using the previous rule is equal to zero, it is impossible to keep the winning strategy and you can take any feasible number of stones $z \in \{1,2,..,\min(k,n)\}$

It is clear that the rule number two defines the optimal strategy while the rule number three describes the stage when we are just waiting for a possible mistake of our opponent and the opportunity to follow the rule number two later on. The strategy for N=26 and k=4 is simply illustrated on the figure number 1. We have deliberately emphasized the groups of five stones that should be taken in one round to guarantee victory for the second player. We can see there that by assuring that k+1=5 stones are removed in each run there will be only one stone left for our opponent in the end of the game.

Having analyzed our version of the game of Nim it is perhaps the right time to determine its complexity and namely the size of the search space. Our version of the game of Nim specified by the total number of stones N and the upper bound k generates N different positions where it is possible to make exactly $m = \min(k,n)$ decisions, where n is the current number of stones on the table. Table number 1 gives a clear idea on how many different strategies are there for the different values of N and k. Although we could see above that this problem can be easily solved by rather simple mathematical deduction, this particular form of it is quite difficult for genetic algorithm because it has no such knowledge about the game. Its search is based on sampling of the relevant search space and from here it is obvious that this game poses a substantial challenge for genetic algorithms.

Figure No. 1. Game of Nim (N=31, k = 4)



GENETIC ALGORITHM IMPLEMENTATION

To represent the problem, we have used the straightforward representation, where individuals are directly encoded as eligible strategies. There are N different positions and that is why the relevant chromosome consists of N genes. The value of the i-th gene (or more precisely its allele) represents in a straight line the decision of the player that should be taken when this particular position within the game has been achieved. To illustrate this encoding easily on a specific example, let us assume that there are 21 stones on the table (N=21) and it is possible to take up to 4 stones at once (k=4). The chromosome

[1, 1, 2, 3, 2, 4, 2, 1, 3, 4, 1, 2, 3, 2, 1, 4, 1, 2, 3, 2, 1]

represents the strategy where the player takes one stone if there is one stone on the table only, he takes one stone if there are two stones there, then two stones providing that there are three stones on the table etc.

We have deliberately chosen the instance of the game where the optimal strategy exists and so it might be worthwhile to show it to supplement the theoretical explanations given above. In this particular case the

optimal strategies for the second player are encoded by the scheme
[1, 1, 2, 3, 4, #, 1, 2, 3, 4, #, 1, 2, 3, 4, #, 1, 2, 3, 4, #].
We can see that the player follows the rule number 2 as we defined it above. The symbols of # stand for arbitrary eligible number of stones, because these four positions within the game relate to the rule number 3. It means that in these cases it is impossible to keep the winning strategy and any decision can be taken. Of course, we can see that if our opponent starts and we play according to this strategy, there is no chance for him to force us to fail.

Table No. 1. Search space sizes for different values of parameters *N* and *k*

Number of stones (N)	Number of strategies (k=4)	Number of strategies (k=5)	Number of strategies (k=6)
1	1	1	1
5	96	120	120
10	98304	375000	933120
15	1,01E+08	1,17E+09	7,26E+09
20	1,03E+11	3,66E+12	5,64E+13
30	1,08E+17	3,58E+19	3,41E+21
40	1,13E+23	3,49E+26	2,06E+29
50	1,19E+29	3,41E+33	1,25E+37

This kind of encoding is a one-to-one encoding, which means that there is only one chromosome for a particular strategy and vice versa. We can also easily see from here that there are 4,123E+11 different strategies in our particular example (*N*=21, *k* = 4) and there are only 256 optimal strategies within them, which poses a reasonable difficulty for any algorithm to be used to discover it.

The fitness of each individual depends on its ability to compete with other strategies and so we have to design a way to measure it. The simplest way to measure this ability is to organize a kind of tournament where winner gets one point while the beaten strategy earns nothing. And now we can see another advantage of the representation we have chosen, because it is very easy to organize a duel between two strategies. A relevant piece of the code written in Prolog is as simple as:

```
duel(Strategy1, Strategy2, Winner) :-
        length_of_chromosome(N),
        % Strategy1 starts
        play1(Strategy1,Strategy2,N,Winner).

play1(Strategy1, Strategy2, N, Winner) :-
        n_th( N, Strategy1, M),
        P is N - M,
        ((P < 1, % it was the last one
          Winner = 2, ! ) ;
        play2(Strategy1,Strategy2,P,Winner)).

play2(Strategy1,Strategy2,N,Winner) :-
        n_th( N, Strategy2, M),
        P is N - M,
        ((P < 1, % it was the last one
          Winner = 1, ! ) ;
        play1(Strategy1,Strategy2,P,Winner)).

    n_th(1, [H|_], H) :- !.
    n_th(N, [_|T], X) :- N > 1,
              N1 is N - 1,
              n_th(N1, T, X).
```

While it is quite simple to solve this problem, we have to cope with the more difficult questions as how many duels should be organized for each individual, and moreover, which individuals should be chosen as its opponents etc. We did some tests based on idea of using random opponents only as in [10] but our results were not satisfactory. That is why we have dwelled on the selection strategies devised in [8] where they examined various approaches in order to speed up the process of evolution. There have been three basic strategies considered in [8]:

1. Entire population is used to evaluate an individual.
2. Random sample of in advance specified number of individuals is used.
3. So called “hall of fame” approach that allows to save specific individuals for testing purposes.

We tested all of them and finally we have employed a kind of hall of fame approach where the best individuals from the former generations are used to evaluate new individuals. The number of individuals in hall of fame clearly depends on the size of population and we managed to get very good results when the size of hall of fame was approximately about one third of the size of the whole population. As the size of our population was kept constant at 400 individuals for all tests reported here we have employed 150 opponents to evaluate each new individual. The size of elite was kept constant and equal to 20 individuals.

We have employed a single population only instead of two populations of “hosts” and “parasites” and that is why we had to create the hall of fame from the same population. This problem can be solved realizing a simple fact that we are looking for strategy for the second player and when evaluating it the opponent strategies are used as the first player strategies. Each second player strategy can be gradually (if it is good enough) exploited to challenge the newly created strategies and that is why this model can rely on a single population.

Another important problem that comes forward when evaluating individuals and that must be solved is caused by mutual interactions and continuing co-evolution of the individuals. To explain this issue let us suppose that each individual is assessed in such a way that it plays the game with 150 opponent strategies and it gets one point for each victory. It is clear that at the beginning of the evolution process when the individuals are generated randomly the population is full of poor strategies and it is rather easy for a mediocre strategy to win many duels and to acquire lot of points. On the other hand, situation gets tougher later on when better and better strategies are present within the population and it is not as easy for them to beat their opponents. It is a kind of well known "red queen syndrome" that implies that each individual within the population should be improved in order to be evaluated at least as it was evaluated earlier. Of course, these findings are irrelevant for the optimal strategy as it beats the others by definition.

This issue creates some problems especially when we want to preserve a part of former population (exploiting either elitism or a steady-state model of reproduction). We have experimented with two different ideas to cope with the above-mentioned problem.

Our first and very straightforward solution was based on the re-evaluation of the survived part of the old population. This approach is indisputable but it is time consuming. The second solution to the problem utilizes a kind of depreciation of the evaluation of the surviving members of population. We tested several schemes involving various factors as for example the number of generation etc. and we have achieved some very interesting results. On the other hand, it is clear that the set up of the relevant parameters is a purely empirical effort. That is why we have returned to the first approach and the "elite individuals" are re-evaluated by the current set of opponents before being incorporated within a newly created population.

It was alluring to measure the fitness of our individuals more exactly and easily by computing the difference between evaluated strategy and the optimal one, which is well-known in this particular case. Because it would be too easy then and rather unfair we did not use it for this purpose. On the other hand it was interesting to measure it and to monitor the progress of the algorithm using this measure and that is why you can see the number of wrong decisions on the figures 2 and 3 where performance of our algorithm is depicted.

Regarding the genetic operators, the chosen representation scheme allowed us to utilize a traditional two-point crossover that is applied with probability p_c=0.75. We have also employed a mutation operator that changes the value of the relevant gene at random with probability p_m=0.005.

All testing reported within this work was done with GAP, a LPA WIN-PROLOG implementation of genetic algorithms package that we developed earlier and it is fully described in [5]. It facilitates manipulation of binary, integer, floating-point, as well as tree structures representations. The package offers a whole range of selection and reproduction schemes, as well as various genetic operators. Moreover, GAP facilitates quick implementation of the custom-tailored genetic operators and evaluation functions that is very important for experimentation.

MACROMUTATION OPERATOR

Our early experiments made us aware of an important recognition that because of the particular encoding we have chosen and the fact that amongst mediocre strategies the better ones are those where end-game is more developed it might be possible to speed up the process of the optimal strategy evolution. Here we have recalled the genetic programming study of Lang [6] who argued that crossover in a population did not perform nearly as well as macromutation operator that was whimsically nicknamed headless chicken crossover. In headless chicken crossover, only one parent is selected from the current population and an entirely new individual is created randomly. The selected parent is then crossed over with this randomly created individual and the offspring is kept only if it is better than or equal to the parent in fitness. Otherwise, it is discarded. Therefore headless chicken crossover is a form of hill climbing.

Lang claimed that headless chicken crossover was much better than crossover but his study was based on one small problem (the Boolean 3-multiplexer problem). That is why some others (see e.g. [2]) disputed his results maintaining that every machine learning technique has a bias – a tendency to perform better on certain types of problems than on others. While Lang picked only one test problem to show some features of this particular macromutation operator and to overgeneralize them, we did it other way round – we employed the headless chicken operator to improve the genetic algorithm performance on our problem. Because of the appropriate encoding our problem is particularly well suited for such an approach.

We have used it in the stage when the members of elite are incorporated within the newly created population. Each member of this elite is challenged by its offspring created by the headless chicken operator and the better of these two is inserted into the new population. Taking into account the size of elite it is not a time consuming operation. We have not studied the effects of this operator when utilized more widely within the population yet. We do believe that it might be an interesting issue for further research.

RESULTS ACHIEVED

The results of our experiments are summarized in the tables number 2 and 3. We have run series of experiments for N = 11, 21, and 31 stones respectively, while the upper limit on the maximal number of stones to be taken in one move has been kept constant (k=4). We have run the algorithm 100 times for each game configuration and the minimum, maximum, and average number of generations needed to find an optimal strategy is reported there.

From these two tables we can clearly see that the utilization of the headless chicken operator remarkably facilitates the search for the optimal strategy. The figures number 2 and 3 illustrate this positive change and on the figure number 3 we can see the percentage of the headless chicken operator utilization during the evolution process itself. It is clear that its influence is higher in the earlier stages of the evolution process when the population is full of poor and mediocre strategies and it is also the main reason for the acceleration of the process as a whole.

Another issue that is clearly visible from these figures is the problem that from the very beginning there are highly evaluated individuals within each population. As we have discussed above it is caused by poor quality of their opponents.

Table No. 2. The number of generations needed to find optimal strategy (100 test runs – GA without macromutation operator)

Number of stones (N)	11	21	31
Min	4	33	76
Max	25	376	592
Average	13,3	91,0	194,1

Table No. 3. The number of generations needed to find optimal strategy (100 test runs - GA empowered by the macromutation operator)

Number of stones (N)	11	21	31
Min	2	33	79
Max	21	69	184
Average	12,6	54,1	126,6

CONCLUSION

The game of Nim is a simple game that is well suited to the research in this field. It has several useful properties including the clearly defined and easily understood optimal strategy. Our experiments reported here show that genetic algorithms represent a suitable tool to tackle it.

We have proposed and tested a new approach utilizing a single population only. Moreover we have shown that by utilizing the headless chicken macromutation operator we can significantly speed up the evolutionary process and the optimal solution is discovered much earlier. While Jones has been criticized [2] for exaggerating his results on one particular example, we have employed this operator deliberately to the problem where we could see that it is likely to perform efficiently. It is clear that the synergy of the suitable encoding and the exploitation of the fitting macromutation delivers encouraging results. We do believe that our approaches described here can be useful when tackling other problems where the suitable strategy goal is pursued and we are going to explore them further.

Figure No. 2. Performance curves (*N*=31, *k* = 4; GA without utilisation of the macromutation operator)

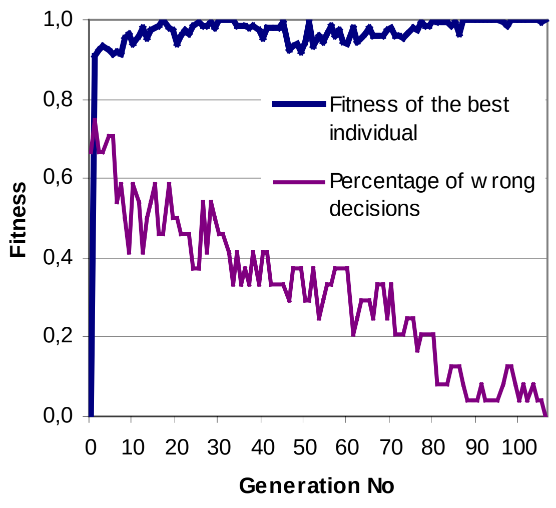
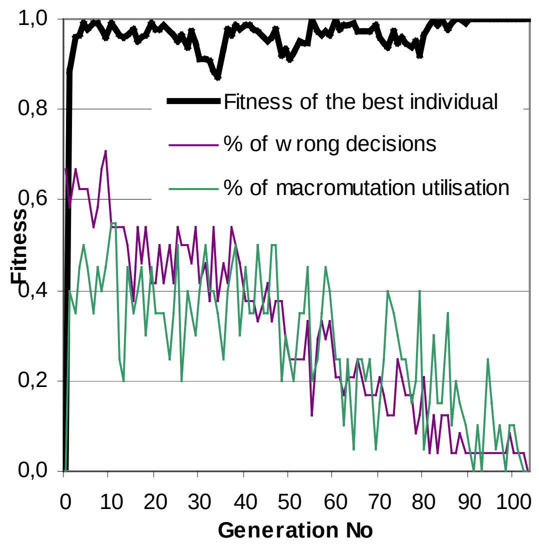


Figure No. 3. Performance curves (*N*=31, *k* = 4; GA empowered by the macromutation operator)



ACKNOWLEDGMENTS

This research has been partially supported by the Grant Agency of the Czech Republic project No.

406/03/0115 and the Czech Ministry of Education project No. MSM 184500001.

REFERENCES

[1] Baba, N., Jain, L. (Eds.): Computational Intelligence in Games. Studies in Fuzziness and Soft Computing, Vol. 62, Springer-Verlag, Heidelberg, 2001.

[2] Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: *Genetic Programming. An Introduction. On the Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann, San Francisco, 1998.

[3] Eiben, A.E., Raué,P.E., Ruttkay, Z.: How to Apply Genetic Algorithms to Constrained Problems. In: Chambers, L. (Ed.): Practical Handbook of Genetic Algorithms. Vol. 1, 1995, pp. 307-353.

[4] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA, 1989.

[5] Hynek, J.: A Prolog Implementation of Genetic Algorithms. Ph.D. Thesis, Charles University, Prague 1998.

[6] Lang, K.J.: Hill climbing beats genetic search on a boolean circuit sznthesis of Koza's. In Proceedings of the Twelfth International Conference on Machine Learning. Tahoe City, CA. Morgan Kaufmann, San Francisco, 1995.

[7] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs (3rd ed.). Springer-Verlag, Berlin 1996.

[8] Rossin, C. D., Belew, R. K.: New Methods for Competitive Coevolution. University of California, San Diego, Department of Computer Science and Engineering, Technical Report #CS96-491, La Jolla 1996.

[9] Rossin, C. D., Belew, R. K.: A Competitive Approach to Game Learning. Proceedings of the Ninth Annual ACM Workshop on Computational Learning Theory, ACM 1996.

[10] Shi, J.: Genetic Algorithms for Game Playing. In: Karr, C. L., Freeman, L. M. (Eds): Industrial Applications of Genetic Algorithms. CRC Press, Boca Raton, 1999, pp. 321-338.