EVOLVING WINNING STRATEGIES FOR NIM-LIKE GAMES

Mihai Oltean
Department of Computer Science,
Faculty of Mathematics and Computer Science,
Babes-Bolyai University, Kogalniceanu 1,
Cluj-Napoca, 3400, Romania.
moltean@cs.ubbcluj.ro

Abstract

An evolutionary approach for computing the winning strategy for Nim-like games is proposed in this paper. The winning strategy is computed by using the Multi Expression Programming (MEP) technique - a fast and efficient variant of the Genetic Programming (GP). Each play strategy is represented by a mathematical expression that contains mathematical operators (such as +, -, *, mod, div, and , or, xor, not) and operands (encoding the current game state). Several numerical experiments for computing the winning strategy for the Nim game are performed. The computational effort needed for evolving a winning strategy is reported. The results show that the proposed evolutionary approach is very suitable for computing the winning strategy for Nim-like games.

Keywords: Nim game, Evolutionary Computation, Genetic Programming, Multi Expression Programming

1. Introduction

Nim is one of the older two-person games known today. While the standard approaches for determining winning strategies for *Nim* are based on the Grundy-Sprague theory (Berlekamp 1982, Conway 1976), this problem can be solved using other techniques. For instance, the first winning strategy for this game was proposed in 1901 by L.C. Bouton from the Harvard University. The Bouton's solution is based on computing the *xor* sum of the numbers of objects in each heap. In other words Bouton computed a relation between the current state of the game and the player which has a winning strategy if it is his/her turn to move.

In this paper, we propose an evolutionary approach for computing the winning strategy for *Nim*-like games. The proposed approach is based on Multi

Expression Programming[1] (Oltean 2003a, Oltean 2003b), which is a fast and efficient alternative to Genetic Programming (GP) (Koza 1992). The idea is to find a mathematical relation (an expression) between the current game state and the winner of the game (assuming that both players do not make wrong moves). The searched expression should contain some mathematical operators (such as +, -, *, *div*, *mod*, *and*, *or*, *not*, *xor*) and some operands (encoding the current game state).

It is widely known (Berlekamp 1982, Gardner 1988) that a winning strategy is based on separation of the game's states in two types of positions: P-positions (advantage to the previous player) and N-positions (advantage to the next player). Our aim is to find a formula that is able to detect whether a given game position belongs to P-positions or to N-positions. Our formula has to return 0 if the given position is a P-position and a nonzero value otherwise. That could be easily assimilated to a symbolic regression (Koza 1992) or a classification task. It is well-known that machine learning techniques (such as Neural Networks or Evolutionary Algorithms (Goldberg 1989) are very suitable for solving this kind of problems. However, the proposed approach is different from the classical approaches mainly because the P and N-positions are usually difficult to be identified for a new game. Instead we propose an approach that checks P and N-position during the traversing of the game tree.

This theory can be easily extended for other games that share several properties with the *Nim* game (i.e. games for which the winning strategy is based on P and N-positions).

The problem of finding N and P-positions could be also viewed as a classification task with two classes. However, we do not use this approach because in this case is required to know the class (P or N) for each game position.

The paper is organized as follows. The *Nim* game is briefly described in section 2. MEP technique is briefly described in section 3. The fitness assignment process is in detail described in section 3.4. Several numerical experiments are performed in section 4. Section 5 outlines the possibility of using the proposed technique for discovering winning strategies for other games.

2. Basics on Nim Game

Nim is one of the oldest and most engaging of all two-person mathematical games known today (Berlekamp 1982, Conway 1976). The name and the complete theory of the game were invented by the professor Charles Leonard Bouton from Harvard University about 100 years ago.

Players take turns removing objects (counters, pebbles, coins, pieces of paper) from heaps (piles, rows, boxes), but only from one heap at a time. In the normal convention the player who removes the last object wins.

The usual practice in impartial games is to call a hot position (N-position - advantage to the next player, i.e. the one who is about to make a move) and a cold one (P-position - advantage to the previous player, i.e. the one who has just made a move).

In 1930, R. P. Sprague and P. M. Grundy developed a theory of impartial games in which *Nim* played a most important role. According to the Sprague-Grundy theory every position in an impartial game can be assigned a Grundy number which makes it equivalent to a *Nim* heap of that size. The Grundy number of a position is variously known as its *Nim-heap* or *nimber* for short (Berlekamp 1982, Conway 1976).

A P-position for the *Nim* game is given by the equation:

x_1 *xor* x_2 *xor* ... *xor* x_n = 0,

where n is the number of heaps, x_i is the number of objects in the i^{th} heap and *xor* acts as the **modulo** 2 operator.

3. Multi Expression Programming Technique

The MEP representation, evolutionary scheme and the fitness assignment process are briefly described in this section.

3.1 Individual Representation

MEP representation is similar to the way in which *C* or *Pascal* compilers translate mathematical into machine code (Oltean 2003). *Pascal* or *C* compilers use so called three addresses code (or intermediary code) when they translate an infix form mathematical expression (i.e. the human readable form) into an executable machine code.

MEP genes are substrings of variable length. Number of genes in a chromosome is constant and it represents *chromosome length*. Each gene encodes a terminal or a function symbol. A gene encoding a function includes pointers towards the function arguments. Function parameters always have indices of lower values than the position of that function itself in the chromosome.

According to the proposed representation scheme the first symbol of the chromosome must be a terminal symbol.

Example

An example of chromosome C_{MEP} is given below:

A representation where numbers on the left positions stand for gene labels (or memory addresses) it is used. Labels do not belong to the chromosome. They are provided for explanation purposes only.

1: a
2: b
3: + 1, 2
4: c
5: d
6: * 3, 5

When MEP individuals are translated into (expressions) computer programs they are read downstream starting with the first position. A terminal symbol specifies a simple expression. A function symbol specifies a complex expression (formed by linking the operands specified by the argument positions with the current function symbol).

For instance, genes 1, 2, 4 and 5 in previous example encode simple expressions composed of a single terminal symbol. The expressions associated to the genes 1, 2, 4 and 5 are:

$E_1 = a$,
$E_2 = b$,
$E_4 = c$,
$E_5 = d$,

Gene 3 indicates the operation + on the operands located at positions 1 and 2 of the chromosome. Therefore gene 3 encodes the expression:

$E_3 = a + b$.

Gene 6 indicates the operation * on the operands located at positions 3 and 5. Therefore gene 6 encodes the expression:

E_6 = $(a + b)$ * d.

The expression encoded into a MEP chromosome is the expression encoded by its last gene. Thus, the expression encoded by the previously described MEP chromosome is $(a + b)$ * d.

3.2 MEP for Evolving Nim Strategies

In order to use MEP technique for evolving a formula that may be used for identifying P and N-positions we have to define the terminal symbols and the function symbols. Note that these sets strongly depend on the problem being solved. For instance if we tackle the *Nim* game, the set T may consists of the following values: number of heaps (n) and the number of objects in each heap.

Thus $T = \{n, a_1, a_2, \ldots, a_n\}$. The set F is more general and it usually contains some mathematical operators. In this paper we use the set F = {+, -, *, *div*, *mod*, *and*, *or*, *xor*, *not*}.

Remarks :

i) For the *Nim* game only the operator *xor* is needed. However, to avoid biases an extended set of operators has been used.
ii) *and*, *or*, *xor* and *not* are interpreted as bitwise operators.

3.3 Genetic Operators

Genetic operators that may be used in conjunction with the MEP technique are one-point crossover and mutation.

One-point crossover

By applying recombination operator, one crossover point is randomly chosen and the parents exchange the sequences after the crossover point.

Mutation

Each MEP gene may be subject of mutation. To preserve the consistency of the chromosome its first gene must encode a terminal symbol. For other genes there is no restriction in symbols changing.

If the current gene encodes a terminal symbol it may be changed into another terminal symbol or into a function symbol. In the last case the positions indicating the function arguments are also generated by mutation.

If the current gene encodes a function the gene may be mutated into a terminal symbol or into another function (function symbol and pointers towards arguments).

3.4 Fitness Assignment Process

In this section the procedure used for computing the quality of a chromosome is described.

Even if this problem could be easily handled as a classification problem (based on a set of fitness cases), we do not use this approach since for the new games it is difficult to find which the P-positions and N-positions are. Instead we employ an approach based on the traversing the game tree. Each nod in this tree is a game configuration (state).

There are three theorems that run the winning strategy for the *Nim* game (Berlekamp 1988):

(i) any move applied to a P-position turns the game into a N-position,

(ii) there is at least one move that turns the game from a N-position into a P-position,

(iii) the final position (when the game is over) is a P-position.

The value of the expression encoded into a MEP chromosome is computed for each game state. If the obtained value is 0, the corresponding game state is considered as being a P-position, otherwise the configuration is considered as being a N-position.

The fitness of a chromosome is equal to the number of violations of the above described rule that arises in a game tree. Thus, if the current formula (chromosome) indicates that the game state encoded into a node of the game tree is a P-position and (the same current formula indicates that) all the game states encoded in the offspring nodes are also P-positions means that we have a violation of the rule b).

Since we do not want to have violations of the previously described rule, our chromosome must have the fitness equal to zero. This means that the fitness has to be minimized.

For a better understanding of the fitness assignment process we provide an example where we shall compute by hand the fitness of a chromosome.

Consider the game state (2,1), and a MEP chromosome encoding the expression $E = a_1 - a_2$*a_1. The game tree of the *Nim* game is given in Figure 3.4.

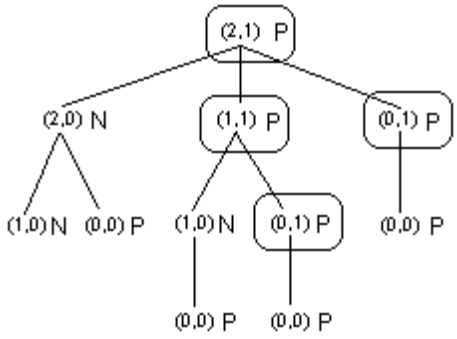

Figure 1. The game tree for a Nim game that starts with the configuration (2, 1). At the right side of each game configuration is printed the configuration' state (P-position or N-position) as computed by the formula $E = a_1 - a_1$*a_2. The configurations that violate one of the three rules described above are encircled.

Figure 3.4 shows that the fitness of a MEP chromosome encoding the formula $E = a_1 - a_2 * a_1$ is four (there are four violations of the winning strategy rules).

3.5 The Evolutionary Algorithm

MEP uses a steady-state (Syswerda 1989) evolutionary scheme. Initial population is randomly generated. The following steps are repeated until a termination criterion is reached: Two parents are selected (from 4 individuals) using binary tournament and are recombined in order to obtain two offspring. The offspring are considered for mutation. The best offspring replaces the worst individual in the current population if the offspring is better than the worst individual.

4. Numerical Experiments

Several numerical for evolving winning strategies for *Nim*-like games are performed in this section.

The purpose of these experiments is to evolve a formula capable to distinguish between a N-position and a P-position for the *Nim* game. We shall analyze the relationships between the success rate and the population size, the chromosome length and the number of generations used during the search process.

In all the experiments it is considered the following configuration for the *Nim* game: (4, 4, 4, 4). This configuration has been chosen in order to have a small computational time. However, this configuration has proved to be enough for evolving a winning strategy.

The total number of game configurations is 70 (which can be obtained either by counting nodes in the game tree or by using the formula of combinations with repetitions). Two permutations of the same configuration are not considered different.

General parameter setting for the MEP algorithm are given in Table 4.

Table 1. General paramters for the MEP algorithm.

Parameter	**Value**
Chromosome length	15 genes
Number of generations	100
Crossover probability	0.9
Mutations	2 mutations / chromosome
Selection strategy	binary tournament
Terminal set	$T_{Nim} = \{n, a_1, a_2, \ldots, a_n\}$.
Function set	$F = \{+, -, *, div, mod, and, not, xor, or\}$

Remark: The success rate is computed by using the formula:

$$\text{Success rate} = \frac{\text{the number of successful runs}}{\text{the total number of runs}}.$$

Experiment 1

In the first experiment the relationship between the population size and the success rate is analyzed. Each MEP chromosome has 15 genes and the search process was evolved for 100 generations. Other MEP parameters are given in Table 4.

The results of this experiment are depicted in Figure 1.

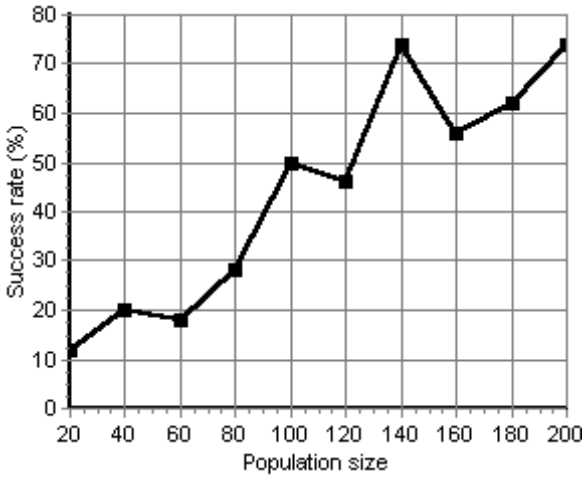

Figure 2. The relationship between the population size and the rate of success. The results are averaged over 50 runs. The population size varies between 20 and 200.

Figure 1 shows that the success rate increases as the population size increases. The highest value - 37 successful runs (out of 50) - is obtained with a population containing 140 individuals. Even a population with 20 individuals is able to yield 6 successful runs (out of 50).

Experiment 2

In the second experiment the relationship between the number of generations and the success rate is analyzed. A population with 100 individuals each having 15 genes is used in this experiment. Other MEP parameters are given in Table 4.

The results of this experiment are depicted in Figure 2.

From Figure 2 it can be seen that MEP is able to find a winning strategy for the *Nim* game in most of the runs. In 41 runs (out of 50) a perfect solutions

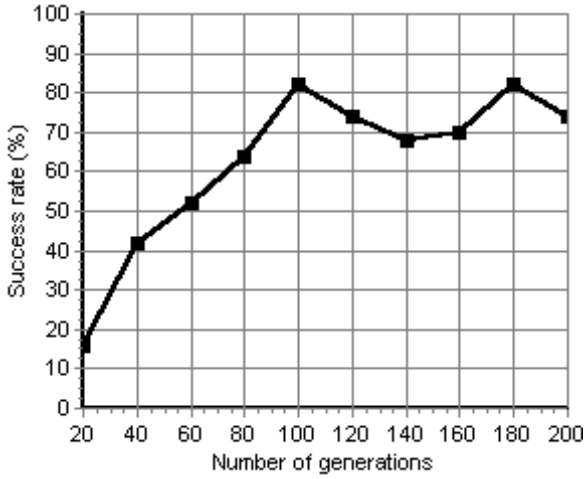

Figure 3. The relationship between the number of generations and the rate of success. The results are averaged over 50 runs. The number of generations varies between 20 and 200.

was obtained after 100 generations. 9 successful runs were obtained when the algorithm is run for 20 generations.

Experiment 3

In the third experiment the relationship between the chromosome length and the success rate is analyzed. A population with 100 individuals is evolved for 50 generations. Other MEP parameters are given in Table 4. The results of this experiment are depicted in Figure 3.

Figure 3 shows that the optimal number of genes of a MEP chromosome is 35. With this value 25 runs (out of 50) were successful. It is interesting to note that the formulas evolved by MEP are sometimes different from the classical a_1 *xor* a_2 *xor* a_3 *xor* a_4. For instance a correct formula evolved by MEP is:

$F = a_1$ *xor* a_2 *xor* $a_3 - a_4$.

This formula is also correct due to the properties of the *xor* operator.

5. Conclusions

In this paper, an evolutionary approach for the *Nim* game has been proposed. The underlying evolutionary technique is *Multi Expression Programming* - a very fast and efficient *Genetic Programming* variant. Numerical experiments have shown that MEP is able to discover a winning strategy in most of the runs.

The proposed method can be easily applied for games whose winning strategy is based on P and N-positions. The idea is to read the game tree and

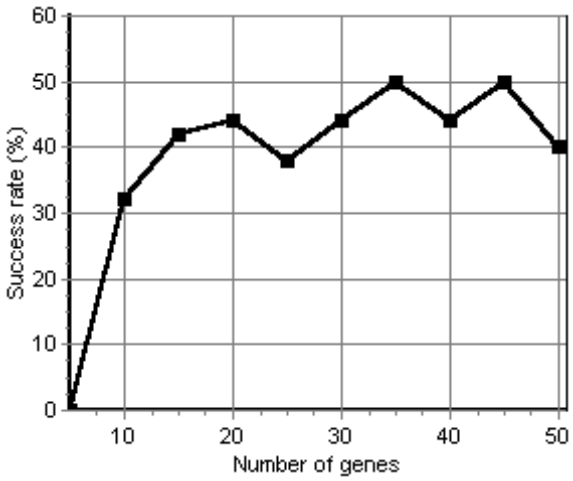

Figure 4. The relationship between the chromosome length and the success rate. The results are averaged over 50 runs. The chromosome length varies between 5 and 50.

to count the number of configurations that violates the rules of the winning strategy.

Notes

1. MEP source code is available from www.mep.cs.ubbcluj.ro.

References

Berlekamp E. R., Conway J. H. and Guy R. K. (1982): *Winning Ways for Your Mathematical Plays*, Academic Press, London.

Conway J. H. (1976): *On Numbers and Games*, Academic Press, London.

Fraenkel S. (1996): *Scenic Trails Ascending from Sea-Level Nim to Alpine Chess*, Games of No Chance, MSRI Publications, Vol. 29.

Gardner M. (1988): *Hexaflexagons and Other Mathematical Diversions*, The University of Chicago Press.

Goldberg D. E. (1989): *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading,MA.

Koza J. R. (1992): *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA.

Oltean M. and Grosan C. (2003a): *Evolving Evolutionary Algorithms using Multi Expression Programming*, The 7^{th} European Conference on Artificial Life, Edited by W. Banzhaf (et al), LNAI 2801, pp. 651-658, Springer-Verlag, Berlin.

Oltean M. (2003b): *Solving Even-Parity Problems using Multi Expression Programming*, The 7^{th} Joint Conference on Information Sciences, Edited by Chen K. (et. al), pp. 295-298.

Syswerda G. (1989) *Uniform Crossover in Genetic Algorithms*, 3^{rd} International Conference on Genetic Algorithms, Edited by Schaffer J.D., Morgan Kaufmann Publishers, pp. 2-9.