Solving Even-Parity Problems using Multi Expression Programming

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

Abstract

In this paper, the Multi Expression Programming (MEP) technique is used for solving even-parity problems. Numerical experiments show that MEP outperforms Genetic Programming (GP) with more than one order of magnitude for the considered test cases.

1. Introduction

Multi Expression Programming (MEP) [5, 6] is a new and very efficient technique that may be used for solving difficult real-world problems. A unique feature of MEP is its ability of storing multiple solutions in a single chromosome [5]. As shown in [5], this feature does not increase the complexity of the decoding process when compared to other Genetic Programming (GP) variants that stores a single solution in a chromosome.

MEP technique has been used [5] for solving symbolic regression problems. Numerical experiments proved that MEP significantly outperforms similar techniques.

In this paper, MEP is used for solving even-parity problems. According to [3] the Boolean even-parity functions appear to be the most difficult Boolean functions to detect via a blind random search. Due to this reason the ability of the evolutionary algorithms of performing an efficient search in the space of solutions can be tested using this problem as a benchmark.

The results of the numerical experiments are compared to those provided by the Genetic Programming techniques [3, 4]. It can be easily seen that MEP outperforms GP with more than one order of magnitude.

2. MEP Technique

In this section *Multi Expression Programming* (MEP) paradigm is briefly described.

2.1. MEP Algorithm

Standard MEP algorithm uses steady state [7] as underlying mechanism. MEP algorithm starts by creating a random population of individuals. The following steps are repeated until a given number of generations is reached. Two parents are selected using a selection procedure. The parents 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.

The algorithm returns as its answer the best expression evolved along a fixed number of generations.

2.2. MEP Representation

MEP genes are (represented by) substrings of a variable length. The number of genes per chromosome is constant. This number defines the length of the chromosome. Each gene encodes a terminal or a function symbol. A gene encoding a function includes pointers towards the function arguments. Function arguments always have indices of lower values than the position of that function in the chromosome.

This representation is similar to the way in which ***C*** and ***Pascal*** compilers translate mathematical expressions into machine code [1].

The proposed representation ensures that no cycle arises while the chromosome is decoded (phenotypically transcripted). According to the proposed representation scheme the first symbol of the chromosome must be a terminal symbol. In this way only syntactically correct programs (MEP individuals) are obtained.

Example

We use a representation where the numbers on the left positions stand for gene labels. Labels do not belong to

the chromosome, they being provided only for explanation purposes.

For this example we use the set of functions $F = \{+, *\}$, and the set of terminals $T = \{a, b, c, d\}$. An example of chromosome using the sets F and T is given below:

1: *a*
2: *b*
3: + 1, 2
4: *c*
5: *d*
6: + 4, 5
7: * 3, 6

2.3. Decoding MEP Chromosome and Fitness Assignment Process

In this section it is described the way in which MEP individuals are translated into computer programs and the way in which the fitness of these programs is computed..

This translation is achieved by reading the chromosome top-down. A terminal symbol specifies a simple expression. A function symbol specifies a complex expression obtained by connecting the operands specified by the argument positions with the current function symbol.

For instance, genes 1, 2, 4 and 5 in the previous example encode simple expressions formed by a single terminal symbol. These expressions 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 4 and 5. Therefore gene 6 encodes the expression:

$E_6 = c + d$.

Gene 7 indicates the operation * on the operands located at position 3 and 6. Therefore gene 7 encodes the expression:

$E_7 = (a + b) * (c + d)$.

E_7 is the expression encoded by the whole chromosome.

There is neither practical nor theoretical evidence that one of these expressions is better than the others. Moreover Wolpert and McReady [8] proved that we cannot use the search algorithm's behavior so far for a particular test function to predict its future behavior on that function. This is why each MEP chromosome is allowed to encode a number of expressions equal to the chromosome length. Each of these expressions is considered as being a potential solution of the problem.

The value of these expressions may be computed by reading the chromosome top down. Partial results are computed by dynamic programming [2] and are stored in a conventional manner.

As MEP chromosome encodes more than one problem solution, it is interesting to see how the fitness is assigned.

Usually the chromosome fitness is defined as the fitness of the best expression encoded by that chromosome.

For instance, if we want to solve symbolic regression problems the fitness of each sub-expression E_i may be computed using the formula:

$$f(E_i) = \sum_{k=1}^{n} \left| o_{k,i} - w_k \right|,$$

where $o_{k,i}$ is the obtained result by the expression E_i for the fitness case k and w_k is the targeted result for the fitness case k. In this case the fitness needs to be minimized.

The fitness of an individual is set to be equal to the lowest fitness of the expressions encoded in chromosome:

$$f(C) = \min_i f(E_i).$$

When we have to deal with other problems we compute the fitness of each sub-expression encoded in the MEP chromosome and the fitness of the entire individual is given by the fitness of the best expression encoded in that chromosome.

2.4. Search Operators

Search operators used within MEP algorithm are crossover and mutation. Considered search operators preserve the chromosome structure. All offspring are syntactically correct expressions.

Crossover

By crossover two parents are selected and are recombined. For instance, within the uniform

recombination the offspring genes are taken randomly from one parent or another.

Mutation

Each symbol (terminal, function of function pointer) in the chromosome may be target of mutation operator. By mutation some symbols in the chromosome are changed. To preserve the consistency of the chromosome its first gene must encode a terminal symbol.

3 Numerical Experiments with Even-Parity Problems

The Boolean even-parity function of k Boolean arguments returns **T** (**True**) if an even number of its arguments are **T**. Otherwise the even-parity function returns **NIL** (**False**) [3].

In applying MEP to the even-parity function of k arguments, the terminal set T consists of the k Boolean arguments d_0, d_1, d_2, ... d_{k-1}. The function set F consists of four two-argument primitive Boolean functions: AND, OR, NAND, NOR. According to [3] the Boolean even-parity functions appear to be the most difficult Boolean functions to detect via a blind random search.

The set of fitness cases for this problem consists of the 2^k combinations of the k Boolean arguments. The fitness of an MEP chromosome is the sum, over these 2^k fitness cases, of the Hamming distance (error) between the returned value by the MEP chromosome and the correct value of the Boolean function. Since the standardized fitness ranges between 0 and 2^k, a value closer to zero is better (since the fitness is to be minimized).

Several numerical experiments with MEP have been performed for solving the even-3-parity, the even-4-parity and the even-5-parity problems.

Two statistics are of high interest:

i. the relationship between the success rate and the number of genes in a MEP chromosome,

ii. the relationship between the success rate and the size of the population used by the MEP algorithm.

3.1. Even-3-Parity and Even-4-Parity Problems

For the even-3-parity problem the terminal set is $T_3 = \{d_0, d_1, d_2\}$. For the even-4-parity problem the terminal set is $T_4 = \{d_0, d_1, d_2, d_3\}$.

Other MEP parameters are given in Table 1.

Parameter	**Value**
Number of Generations	51
Mutation probability	0.1
Crossover type	Uniform
Selection	q-tournament (q = 10% of the Population Size)
Terminal set	$T_3 = \{d_0, d_1, d_2\}$ $T_4 = \{d_0, d_1, d_2, d_3\}$
Function set	F = {AND,OR,NAND,NOR}

Table 1. The general parameters of the MEP algorithm for solving even-parity problems.

A population of 100 individuals is used when the influence of the number of genes is analysed and a code length of 200 genes is used when the influence of the population size is analysed.

For reducing the chromosome length we keep all the terminals on the first positions of the MEP chromosomes. We also increased the selection pressure by using larger values (usually 10% of the population size) for the tournament sample.

The results are depicted in Figures 1 and 2.

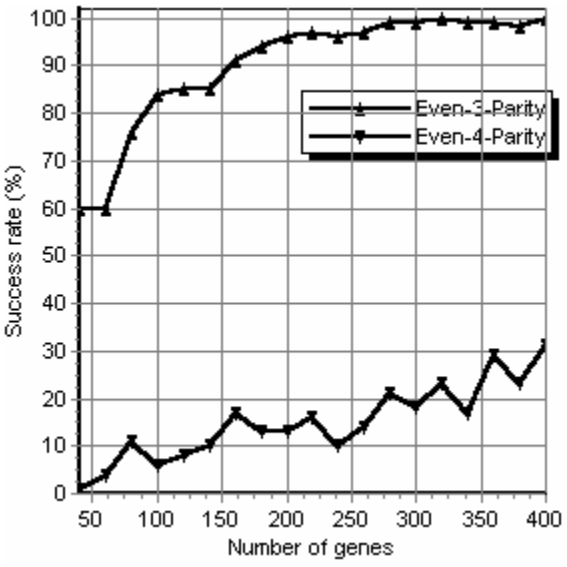

Figure 1: The relationship between the success rate and the number of genes in a MEP chromosome. Results are summed over 100 runs.

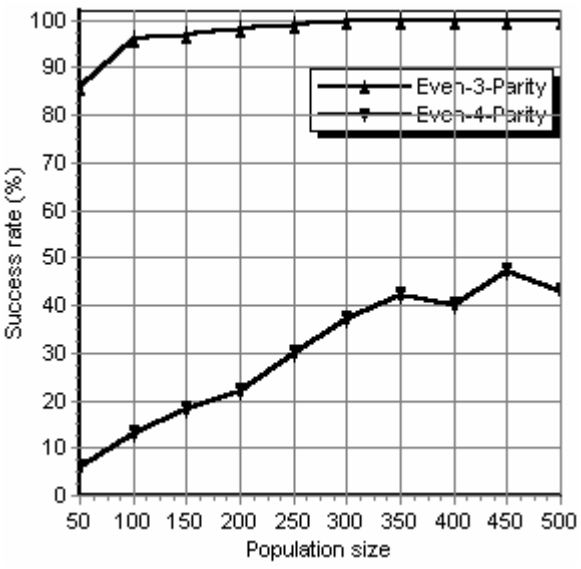

Figure 2: The relationship between the success rate and the population size employed by the MEP algorithm. Results are summed over 100 runs.

From Figures 1 and 2 it can be seen that MEP performs very well on the considered test problem.

GP technique (without Automatically Defined Functions (ADFs)) has been used for solving the even-3 and even-4 parity problems using a population of 4000 individuals [3]. The cumulative probability of success was 100% for the even-3-parity problem and 42% for the even-4-parity problem [3].

A perfect comparison between MEP and GP cannot be made due to the incompatibility of the respective representations. Having this in mind we do provide a raw comparison between MEP and GP.

From Figure 1 it can be seen that MEP needs a population of only 100 individuals with 270 genes to achieve a success rate of 100% for the even-3-parity problem. From Figure 2 it can be seen that a population of only 300 individuals having 200 genes are enough to solve the even-3-parity problem. Knowing that GP used a population of 4000 individuals to achieve a success rate of 100% we may infer that MEP needs a population smaller with more than one order of magnitude than the population needed by GP to solve the even-3-parrity problem. For the even-4-parity problems MEP reaches a success rate of 42% with a population of 350 individuals (as it can be seen from Figure 2). Again we deal with an improvement of one order of magnitude.

3.2. Even 5-Parity Problem

For this problem MEP was run with a population of 1000 individuals having 600 genes each. In 5 runs (out of 30) MEP was able to find a perfect solution for this problem, yielding a success rate of 16.66%. Note that for this problem GP (without ADFs) was not able to obtain a solution (within 20 runs) even with a population of 4000 individuals [3]. When the population size was increased to 8000 individuals a solution was obtained by GP after 8 runs [3].

We may conclude that MEP significantly outperforms standard GP (without ADFs) for these particular cases of the even-parity problem.

4. Conclusions and Further Work

In this paper, MEP technique has been used for solving even-parity problems. Numerical experiments have shown that MEP significantly outperforms standard GP on the considered examples.

Further research will be focused on developing an Automatically Defined Functions (ADFs) [3] system for MEP. It is expected that using ADFs will improve the performance of the MEP technique in the same way it has improved the performance of the GP technique.

References

[1] A. Aho, R. Sethi and J. Ullman, “Compilers: Principles, Techniques, and Tools”, Addison Wesley, 1986.

[2] R. Bellman, “Dynamic Programming”, Princeton, Princeton University Press, New Jersey, 1957.

[3] J. R. Koza, “Genetic Programming: On the Programming of Computers by Means of Natural Selection”, MIT Press, Cambridge, MA, 1992.

[4] J. R. Koza, “Genetic Programming II: Automatic Discovery of Reusable Programs”, MIT Press, Cambridge, MA, 1994.

[5] M. Oltean and D. Dumitrescu, “Multi Expression Programming”, Technical-Report, UBB-01-2002, (available at www.mep.cs.ubbcluj.ro).

[6] M. Oltean and C. Groşan, “Evolving Evolutionary Algorithms using Multi Expression Programming”, European Conference on Artificial Life, Dortmund, 14-17 September 2003, Accepted for publication.

[7] G. Syswerda, “Uniform Crossover in Genetic Algorithms”, Proceedings of the 3rd International Conference on Genetic Algorithms, J.D. Schaffer (Editor), Morgan Kaufmann Publishers, CA, pp 2-9, 1989.

[8] D.H. Wolpert and W.G. McReady,“No Free Lunch Theorems for Optimisation”, IEEE Transaction on Evolutionary Computation, Vol.1, pp 67-82, 1997.