BABES-BOLYAI University of Cluj-Napoca
Faculty of Mathematics and Computer Science
Proceedings of the Symposium **"Colocviul Academic Clujean de INFORMATICA",** p. 93-98

Designing a Component-Based Machine Using Multi Expression Programming

A. Fanea[1] and L. Dioşan[2]

Abstract. *A new method for designing component-based machine using Multi Expression Programming MEP is introduced. The elements of MEP are presented. The mapping between component-based systems and MEP are presented.*

Keywords: Component-Based System, Multi Expression Programming

1. Introduction

The main advantage of component-based system development is the reuse of components when building applications. Instead of developing a new system from scratch, already existing components are assembled to give the required result. To incorporate a component in a system successfully, a procedure of selection, composition and integration, and finally, test and verification must be followed.

In [C1] the processes of component composition and integration are described. In [F1] a formal model for component composition is described. If we consider that a compound component is formed from two simple components then there are two basic ways in which these simple components can depend on each other: parallel and serial composition.

The two basic ways – see figure 1- in which these simple components can depend on each other are:

- **a parallel composition**, **A || B**, in which the operations performed on data are independent and there is not dependency between outports(A) and outports(B);
- **a serial composition**, **A + B,** in which the **B** component expects some results from component **A**

More information about parallel and serial composition can be found in [M1].

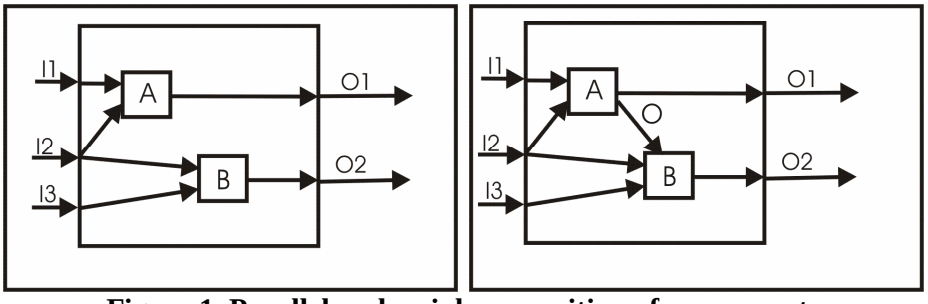

Figure 1. Parallel and serial composition of components

Any compound component can be described with these two basic operations, no matter how many simple components it contains. Compound components are built from atomic components

[1] [2] "Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, Department of Computer Science, {*afanea,lauras*}*@cs.ubbcluj.ro*

(source, destination or simple components) using parallel composition and serial composition, depending on the interdependencies inside a compound component. Having specified the simple components that we need to compose in order to obtain the compound component, we give a model [F1] to obtain all the possibilities of using parallel and serial composition to have at the end the black box that we want.

2. Multi Expression Programming (MEP)

Basic ideas. *Multi Expression Programming* (MEP) [O2] is a Genetic Programming (GP) [O4, K1, K2, L1] variant that uses a linear representation of chromosomes. MEP individuals are strings of genes encoding complex computer programs. A unique MEP feature is its ability of encoding multiple solutions of a problem in a single chromosome.

Algorithm. The five major preparatory steps for the basic version of the problem to the genetic programming require the human user to specify:

- set of terminals;
- set of primitive functions;
- fitness measure;
- certain parameters for controlling the execution;
- termination criterion and method for designing the result of the execution.

Standard MEP algorithm uses steady-state evolutionary model [S1] as its underlying mechanism. The initial population is randomly chosen.

The following steps composes the main mechanism for MEP algorithm:

- Selection – binary tournament;
- 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;
- Replace the worst individual in current population with the best offspring (if the best offspring is better than the worst individual);
- These steps are repeated until a given generation number is reached.

The standard MEP [O3] is outlined below:

S1. Randomly create the initial population *P*(0)
S2. For *t* = 1 to *Max Generations* do
S3. For *k* = 1 to | P(*t*) | / 2 do
S4. *p*1 = *Select*(*P*(*t*)); // select one individual from the current population
S5. *p*2 = *Select*(*P*(*t*)); // select the second individual
S6. *Crossover* (*p*1, *p*2, *o*1, *o*2); // crossover the parents (p1, p2), offspring (o1, o2)
S7. *Mutation* (*o*1); // mutate the offspring o1
S8. *Mutation* (*o*2); // mutate the offspring o2
S9. if *Fitness*(*o*1) < Fitness(*o*2) then
S10. if *Fitness*(*o*1) < the fitness of the worst individual in the current population
S11. then *Replace* the worst individual with *o*1;
S12. else if *Fitness*(*o*2) < the fitness of the worst individual in the current population
S13. then *Replace* the worst individual with *o*2;
S14. Endfor
S15. Endfor

The steady-state MEP algorithm starts with a randomly chosen population of individuals.

The following steps are repeated until a termination condition is reached. Two parents are selected using a selection procedure and they are recombined in order to obtain two offspring. The offspring are mutated and the best of them replaces the worst individual in the current population (if the offspring is better than the worst individual in the current population).

Representation. The MEP genes are (represented by) substrings of variable length. The number of genes in a chromosome is constant and it represents the chromosome length. Each gene encodes a terminal (an element in the terminal set *T*) or a function symbol (an element in the function set *F*) [F2]. 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 in a chromosome must be a terminal symbol. In this way only syntactically correct programs are obtained.

Examples. In the following representation 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*	6: *c*
2: *b*	7: + 5, 6
3: - 1, 2	8: *d*
4: *a*	9: + 7, 8.
5: + 3, 4	

The translation of the MEP individuals into computer program is achieved by reading the chromosome top-down. A terminal symbol specifies a simple expression and a function symbol specifies a complex expression obtained by connecting the operands specified by the argument positions with the current function symbol.

Genes 1, 2, 4, 6 and 8 in the previous example encode simple expressions formed by a single terminal symbol. These expressions are E_1=a, E_2=b, E_4=a, E_6=c, E_8=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 5 indicates the operation + on the operands located at positions 3 and 4. Therefore gene 5 encodes the expression E_5 = (a - b) + a.

Gene 7 indicates the operation + on the operands located at position 5 and 6. Therefore gene 7 encodes the expression E_7 = ((a - b) + a) + c.

Gene 9 indicates the operation + on the operands located at position 7 and 8. Therefore gene 9 encodes the expression E_9 = `(((a - b) +a) + c) +d.` E_9 is the expression encoded by the whole chromosome.

3. Applying MEP to Component-Based System

Basic ideas. MEP may be efficiently applied to determine optimal components combination of a component-based system. Having the components involved in the composition for obtaining the complex system, we can apply MEP algorithm to components, where the operations are the two operations for component composition: parallel and serial composition.

Our pattern will be represented as an MEP component-based program whose elements will be compound during the EA evolution. We have chosen Multi Expression Programming for representing the patterns because MEP provides an easy way to store a sequence of instructions (elements). In our approach, MEP will store only one solution (pattern) per chromosome because is difficult to handle multiple evolutionary patterns in the same chromosome. We will still use the MEP notation but there will be only one solution per chromosome in all cases and experiments below.

Algorithm. The steady-state MEP algorithm starts with a randomly chosen population of individuals. The following steps are repeated until a termination condition is reached. Two parents are selected using binary tournament selection and they are recombined in order to obtain two offspring.

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.

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.

Representation. The set of terminals are:

$T = \{c_1, c_2, c_3, c_4, \ldots, c_n\}$, where c_i is the i-th component

and the set of primitive functions:

F = {+, -}, where "+" represents a serial composition and "-" represents a parallel composition.

Fitness measure: we have two criteria to evaluate a chromosome and each criteria has a weight in the final fitness:

- components evaluation (fitness)– 75%, we;
- quality performance (fitness) – 25%, wq.

$$\text{fitness}_e = \prod_{i=1}^{chromoson\_operations\_number} eval(c_i)$$, where n is the number of components;

$$\text{fitness}_q = \bigvee_{k=1} c_i \otimes c_j$$, where i =1,*n*, *j* = 1, *n*, $i \neq j$

fitness = fitness$_e$ * we + (| fitness$_q$ - fitness$_q$*| / fitness$_q$*)* wq

the best chromosome will be that with maximum fitness.

Stop criterion and method for designing the result of run - fitness < ε, where ε is a little value (or it is possible to repeat the above steps until a generation number is reached)

Examples. The example is very simple and didactical: verify if the pair (nextPrime(cd(a, b)), nextPrime(c)) is a pair of twin numbers. (cd =greatest common divisor).

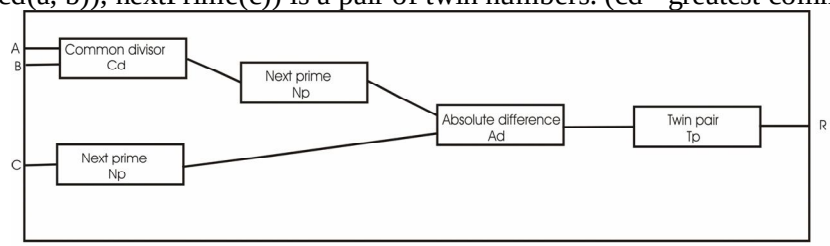

Figure 2. Component-Based System for solving the problem of twin numbers.

For a CBMMEP example it is possible to recall the MEP representation as given in section 2. A MEP chromosome can have the form:

1: *a*	6: *c*
2: *b*	7: + 5, 6
3: - 1, 2	8: *d*
4: *a*	9: + 7, 8.
5: + 3, 4	

We will replace the terminal set *{a, b, ...}* by the new terminal symbol *{np, cd, ad, tp}*[2] which is specific to our purpose. Also the function set *{+, *, -, ...}* will be replaced by *{ ||, + }*.

Example 1. An example of a MEP chromosome encoding a pattern is given below

1: np
2: cd
3: || 1, 2
4: np
5: + 3, 4
6: ad
7: + 5, 6
8: tp
9: +7, 8

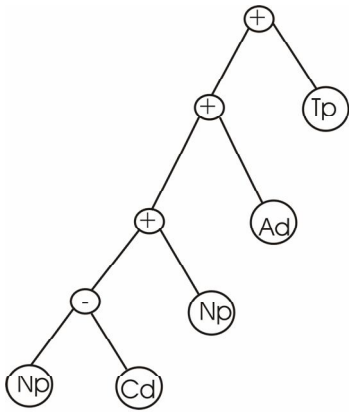

Figure 3. MEP chromosome (e.g. 1)

This MEP chromosome (*(((np || cd) + np) + ad) +tp*) should be interpreted as follows (if the machine has 3 numbers A, B and C as inputs):

1. A serial composition is performed: the greater common divisor of A and B (let be D) and verification if D is prime;
2. A parallel composition is process: verification if C is prime (suddenly with previous two operations);
3. A serial composition is performed: absolute difference between D and C (let be E);
4. A new serial composition: verification if A, B and C formed a twin pair.

Example 2. An example of a MEP chromosome encoding a pattern is given below

1: np
2: cd
3: + 1, 2
4: np
5: + 3, 4
6: ad
7: + 5, 6
8: tp
9: +7, 8

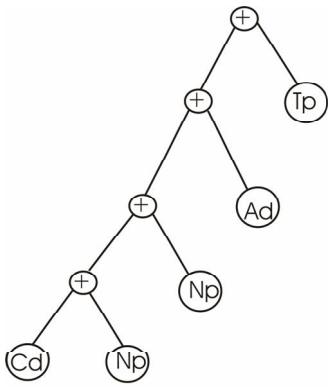

Figure 4. MEP chromosome (e.g. 2)

This MEP chromosome (*(((np + cd) + np) + ad) + tp*) should be interpreted as follows: If the machine has 3 numbers A, B and C as inputs:

1. A serial composition is performed: the greater common divisor of A and B (let be D) and verification if D is prime;
2. A serial composition is performed: verification if C is prime;
3. A serial composition is performed: absolute difference between D and C (let be E);

[2] np – next prime, cd – common divisor, ad – absolute difference, tp – twin pair

4. A new serial composition: verification if (nextPrime(cd(A, B)), nextPrime(C)) formed a twin pair.

4. Conclusions

In this paper, MEP has been used for optimal designing of a component-based machine. In future the model will be extended to components with output wired to more than one component. For this purpose the chromosome tree does not has to be a binary tree.

References

[C1] Ivica Crnkovic, Magnus Larsson, *Building reliable component-based software systems*, 2002

[F1] A. Fanea, S. Motogna, *A Formal Model for Component Composition*, Proceedings of the Symposium "Zilele Academice Clujene", 2004, pp. 160-167

[F2] C. Ferreira, *Gene Expression Programming: a New Adaptive Algorithm for Solving Problems*, Complex Systems, Vol. 13, pp. 87-129, 2001.

[K1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[K2] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, MA, 1994.

[L1] www.genetic-programming.com

[M1] Bazil Parv, Simona Motogna, *A formal model for components*, Studia Univ. VaBES-Bolyai, Informatica, Volume XLVII, Number 1, 2002

[O1] M.Oltean, C. Grosan, *Evolving Digital Circuits using Multi Expression Programming*, NASA/DoD Conference on Evolvable Hardware, 24-26 June, Seattle, Edited by R. Zebulum, D. Gwaltney, G. Horbny, D. Keymeulen, J. Lohn, A. Stoica, pages 87-90, IEEE Press, NJ, 2004.

[O2] M. Oltean, D. Dumitrescu, *Multi Expression Programming*, technical report, Babes-Bolyai University, (available at www.mep.cs.ubbcluj.ro), 2002.

[O3] M. Oltean, *New Evolutionary Computation Models and their Applications to Machine Learning algorithm*, PhD Thesis, 2004

[O4] Oltean, M. (et al.), *Evolving Digital Circuits for the Knapsack Problem*, International Conference on Computational Sciences, E-HARD Workshop, Edited by M. Bubak, G. D. van Albada, P. Sloot, and J. Dongarra, Vol. III, pp. 1257-1264, 6-9 June, Krakow, Poland, Springer-Verlag, Berlin, 2004

[S1] Syswerda, G., *Uniform Crossover in Genetic Algorithms*, Schaffer, J.D., (editor), Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 2-9, Morgan Kaufmann Publishers, San Mateo, CA, 1989.