A Novel Methodology For Diversity Preservation In Evolutionary Algorithms

Giovanni Squillero
Politecnico di Torino - DAUIN
Corso Duca degli Abruzzi 24
10129 Torino - Italy
Tel: +39 011564.7092
giovanni.squillero@polito.it

Alberto P. Tonda
Politecnico di Torino - DAUIN
Corso Duca degli Abruzzi 24
10129 Torino - Italy
Tel: +39 011564.7091
alberto.tonda@polito.it

ABSTRACT

In this paper we describe an improvement of an entropy-based diversity preservation approach for evolutionary algorithms. This approach exploits the information contained not only in the parts that compose an individual, but also in their position and relative order. We executed a set of preliminary experiments in order to test the new approach, using two different problems in which diversity preservation plays a major role in obtaining good solutions.

Categories and Subject Descriptors

I.m [Computing Methodologies]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Evolutionary Algorithms, Diversity Preservation

1. INTRODUCTION

Diversity preservation is one of the main open issues of Evolutionary Computation. Since new individuals of each generation are usually created by applying genetic operators to those already present in the population, it is important to avoid that all the population is filled with identical or nearly identical copies of the same individual. Simplistic approaches propose to avoid individuals with equal fitness. However, these techniques are inadequate where widely different individuals may have a similar fitness value. In the literature, different strategies have been developed to preserve diversity even in these circumstances, such as the island model genetic algorithm, the lattice genetic algorithm and the entropy driven approach.

The island model genetic algorithm, as other coarse-grained mechanisms, is based on considerations from theories of natural evolution and from the efficiencies of parallel computer architectures. Instead of having a single large population, this algorithm uses several distinct subpopulations, which alternate extensive periods of isolated evolution (computation) with occasionally episodes of migration (communication). Since different populations are likely to explore different portions of the solution space, the migration between subpopulations may help mixing genetic information and preserving diversity. This approach, however, leaves some open issues such as dimensioning the subpopulations (which is strongly problem-dependant) and the optimal parameters of the migration. Also, while most empirical results show that island evolutionary algorithms are more efficient than evolutionary algorithms with just one population, this approach seems to be more suitable for separable problems and problems with multiple solution paths.

A different kind of approach is the lattice (or cellular) model, where each individual occupies a cell in a regular lattice (or a more general graph). All interactions, like mating and selection, are local and limited to a neighborhood. Size and shape of the neighborhood are closely related to selection pressure [4]. Since each individual can interact only with its neighbors, this approach allows the survival of individuals with unusual genotypes. Choosing the appropriate neighborhood size and shape is thus critical, and could be problem-related.

In [3] we proposed an alternative approach. While the two latter models focus on limiting the interaction between individuals, the diversity preservation was implemented by conserving individuals with peculiar genotypes despite their fitness values. By considering the population as a message, and each allele appearing in an individual as a symbol, we can compute a measure of the entropy of the message, based on the number of symbols and their occurrences, as in the Shannon entropy formula [6].

$$H(X) = \mathrm{E}(I(X)) \;\; = \;\; -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Figure 1. Shannon Entropy formula.

We defined delta-entropy as the difference between the total entropy of the population and the entropy of the population without the symbols appearing in an individual. Individuals with an unusual genotype would contribute to the message with a large number of unusual symbols, and would thus have great values of delta-entropy. Once those individuals are identified, they could be selected as parents to generate new individuals, thus preserving part of their genotype in the new generations. In this approach, individuals are represented by a directed non-cyclic graph, where each node is an allele of a certain gene. Thus, two individuals (with dissimilar fitness value and structure) could have the same node values (thus the same allele), just in a different order, and

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-131-6/08/07.

both would contribute with the same quality and amount of information to the message. The idea proposed in this paper is to modify the definition of "symbols" connected with an individual, in order to preserve "unique structures" present in the population without altering the entropy computation.

2. A NEW CONCEPT OF ENTROPY

We propose to modify the concept of symbols of a single individual, considering the presence of n-uples of nodes and the position of a certain allele in a graph as atomic element.

A possibly useful information is the order of the nodes of a single individual. Since two distinct positions of the same allele in an individual could have a completely different meaning, we chose to ignore the information related to the allele, focusing instead on both the allele and its position inside the directed graph which describes the individual in µGP3. Instead of considering an allele as a symbol, we coded the information of the value of the instance and its position in a single symbol. By doing so, individuals with recurring alleles of the same gene in a different order have now higher values of delta-entropy. In Figure 3, two different individuals which would contribute to the message with the same symbols, have now only one symbol in common using the new entropy computation.

We also consider valuable to search for n-grams (defined here as n consecutive node values) occurrences in each individual, adding this data to the contribution given to the message by the individual itself. [2] N-grams in the population are coded as symbols, thus letting us discriminate individuals with unusual patterns very easily, without changing the overall entropy-driven approach of the algorithm: even individuals with an atypical order of diffused values of single nodes possess now a high value of delta-entropy, which is an interesting result with regards to preserve diversity in the population. In Figure 2, two different individuals which would contribute to the message with the same symbols using the old entropy computation have only two identical symbols using the new entropy computation.

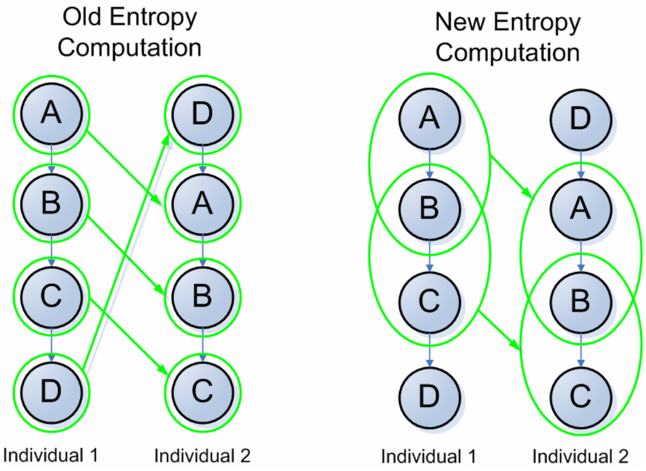

Figure 2. Corresponding symbols in different individuals using old and new entropy computation.

3. EXPERIMENTAL EVALUATION

In order to verify the new entropy approach described in the previous section, we executed a series of preliminary experiments, comparing the results obtained using the new approach with results obtained using the old entropy computation and using no entropy computation at all.

The first benchmark was evolving assembly code in order to obtain a function capable of discerning between prime and non-prime numbers ranging from 1 to 20. The second benchmark was evolving the code used by a simple program to draw a path into a virtual arena, in order to reach a given goal in the least possible number of instructions.

In all the experiments described in the following subsections we used various versions of the evolutionary algorithm known as µGP3 [3]. The version with the new entropy computation, was implemented in order to consider occurrences of n-grams of order 2 and 3.

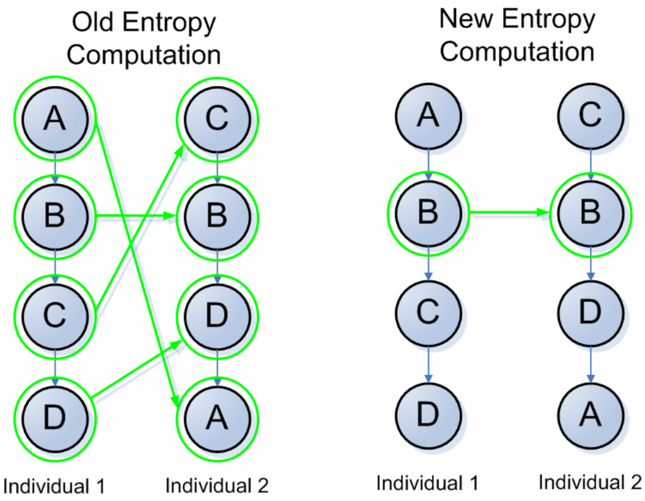

Figure 3. Corresponding symbols in different individuals using old and new entropy computation.

3.1 Prime Numbers Function

In this benchmark, our goal is finding an assembly function which returns 1 whenever it has a prime number passed as an argument, and 0 otherwise. Each individual is a block of assembly code, of arbitrary length. Only a subset of all possible assembly x86 instructions can appear in the code (addl, subl, movl, andl, orl, xorl, test, cmp, ja, jz, jnz, je, jne, jc, jnc, jo, jno) while the operands of each instruction can only be four registers (ax, bx, cx, dx) and integers (with values between 0 and 255). "Jump" instruction can only point to labels after the jump instruction itself, thus preventing endless loops. In this experiment we tested each individual with numbers ranging from 1 to 20.

The evaluation is performed by a script that compiles an individual and then executes a simple C program which invokes the individual as a function multiple times, each time passing a number as an argument and expecting a return value of either 0 or 1. In all the experiments, we used a (µ, λ) strategy with µ = 1000 and λ = 33, and a generational approach with an elitist strategy that preserves the best 3 individuals. Please note that, in µGP3, λ is in fact the number of genetic operators applied to the population at each step, thus the offspring size may be different. In order to select the individuals to pass to the genetic operators, we used a tournament selection (with tournament size of 3). The algorithm was set to stop when the maximum possible fitness value or a given number of generations was reached.

```
            .file       "foo.c"
            .text

.globl foo
            .type       foo, @function
foo:
            pushl       %ebp
            movl        %esp, %ebp
            movl        8(%ebp), %eax
            pushl       %ebx
            pushl       %ecx
            pushl       %edx
            movl        $93, %ebx
            movl        $133, %ecx
            movl        $104, %edx

# prologue
            andl        $225, %eax
                        jne         nFRW
                        xorl        %ecx, %ebx
                        jz          nFRV
                        andl        $32, %ecx
                        addl        %ecx, %ebx
                        jne         nFRV
                        addl        $81, %ecx
                        cmp         %ebx, %ebx
                        addl        %ecx, %eax
                        cmp         %eax, %edx
                        addl        $95, %ecx
                        jno         nFRV
                        addl        %ecx, %ebx
                nFRV:   jne         nFRZ
                nFRW:               orl         %eax, %edx
                        orl         %eax, %ebx
                        subl        %ebx, %ebx
                nFRZ:
#epilogue

            popl        %edx
            popl        %ecx
            popl        %ebx
            popl        %ebp
            ret
            .size       foo, .-foo

            .ident      "GCC: (GNU) 4.1.1 20070105 (Red Hat
4.1.1-51)"
            .section    .note.GNU-stack,"",@progbits
```

Figure 4. An example of individual in the first benchmark.

In the preliminary experiments, we compared the performance of three different tournament selection strategies:
- a selection based only on the fitness values of the chosen individuals
- a selection based on entropy values (with the old entropy computation)
- a selection based on entropy values (with the improved entropy computation we described in section 2).

In all the experiments we found out that the latter approach is the most efficient, obtaining the desired function in 600-700 generations, while the old entropy approach and an experiment carried out with a tournament selection based only on fitness values did not manage to converge even after 2000 generations.

In these experiments we tested only the two extreme cases of tournament selection based either only on the fitness value or on entropy values.

3.2 Finding the Right Path

In this experiment, our goal is to obtain an individual which can arrive to a certain point on the opposite side of an arena without intersecting an obstacle, and doing so in the least possible number of instructions. Each individual is a block of code, between 8 and 50 instructions. There are only two possible instructions: MOVE (which accepts a real parameter ranging from 0.0 to 40.0) and ROTATE (which accepts a rotation expressed in radiant, thus ranging from -π to π).

```
rotate 0.128501499703256
move 16.4006824043443
rotate 0.584000940527606
rotate 0.30873892699125
move 37.7772101402074
move 33.8879641796075
move 37.7759674859092
rotate -1.13261208546878
rotate 0.20164306478673
rotate -0.473967818304483
move 15.9922375998931
move 23.6644477504869
rotate -0.433767526492107
move 35.7594037608348
move 35.7594037608348
move 36.9633255101185
rotate 1.78907118768571
move 1.02390550580111
move 18.6992676756395
rotate 0.30873892699125
move 37.7759674859092
```

Figure 5. An example of individual in the second benchmark.

The evaluation takes the individual and follows its instructions by "moving" into an arena of 250 x 100 units of length, with two obstacles. If the individual intersects an obstacle, its position is not updated and further instructions are not executed.

We have two fitness parameters: one dependent on the distance from the objective, and one dependent on the number of instructions of the individual. Note that each obstacle is a local optimum with regards to the first fitness value: since there is only a small vertical space of 10 units of length where an individual could pass without intersecting an obstacle, it is relatively difficult for an individual to escape the two local optima.

We used a (μ, λ) strategy with μ = λ and population sizes (of 10, 20, 30, and 50 individuals), and a generational approach with an elitist strategy that preserves the best 4 individuals. We executed different runs with the same stopping condition: evaluation of 2 M individuals, in order to compare the best individuals emerged after each run. We used a tournament selection to choose the individuals to pass to the genetic operators:

- in the first case, the selection was based only on the fitness values of the individuals
- in the second, the selection was based on entropy values instead
- a third experiment is running, with an old entropy computation, in order to evaluate the improvement of the new approach

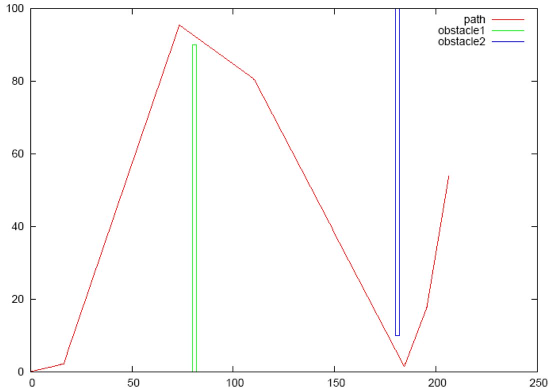

Figure 6. The path produced by the execution of the code provided in Figure 4.

In the preliminary experiments we always obtained an improvement in the fitness values of the best individual using the new entropy approach, for the same population sizes. Again, we used only two kinds of tournament selection, but we are running more tests with various fitness holes and greater population sizes, in order to have a better comparison of the performance of the new entropy computation under different conditions.

Table 1. Results of the preliminary experiments with the second benchmark, comparing the best individual obtained through tournament selection based on fitness values with the best one obtained through tournament selection based on entropy values.

Population Size	Best Individual Length	
	TS based on Fitness	TS based on Entropy
10	NO SOLUTION	22
20	31	26
30	34	23
50	28	23

4. CONCLUSIONS

In this paper we presented a way to improve the approach to preserve diversity in the population used by the evolutionary algorithm µGP. This approach views the population as a message: each individual of the population contributes to the message with a series of symbols. By computing a difference between total entropy of the message and the entropy of the message without the symbols added by an individual, we can obtain a delta-entropy which measures the diversity of the individual itself with regards to the population.

We modified the approach by including as symbols the information on the occurrences of n-grams of values in the nodes composing an individual and on the order of the occurrences of alleles in an individual.

Preliminary experiments demonstrated that the new approach is promising, as it performed more efficiently than the old one in the totality of the tests executed, and much more efficiently when compared to the same tests carried out without any mean to preserve diversity.

5. ACKNOWLEDGMENTS

Our thanks to Massimiliano Schillaci and Alessio Moscatello for developing the evaluators for the benchmark presented, and to Danilo Ravotto and Ernesto Sanchez for constructive criticisms and invaluable advice.

6. REFERENCES

[1] Martin, W.N. , Lienig J. and Cohoon J.P. *Island (migration) models: evolutionary algorithms based on punctuated equilibria*, In Handbook of Evolutionary Computation, IOP Publishing Ltd and Oxford University, 1997.

[2] Tomassini, M. and Giacobini, M. *Spatial and Temporal Dimensions in Evolutionary Systems,* GSICE05, 2005.

[3] Corno F., Sanchez E., Squillero G., *Evolving Assembly Programs: How Games Help Microprocessor Validation,* IEEE Transactions on Evolutionary Computation, Special Issue on Evolutionary Computation and Games, Dec. 2005, vol. 9, pp. 695-706

[4] Sarma J., De Jong K., *An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms,* PPSN IV (Berlin, 1996)

[5] E. Burke, S. Gustafson, G. Kendall, *Diversity in Genetic Programming: An Analysis of Measures and Correlation With Fitness,* IEEE Transactions on Evolutionary Computation, Feb 2004, Vol 8, No I, pp 47-62

[6] Shannon C.E., *A Mathematical Theory of Communication*, Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October, 1948