Cooperative co-evolution inspired operators for classical GP schemes

Malek Aichour[1] and Evelyne Lutton[2]

[1] INRIA Rocquencourt - Domaine de Voluceau, BP 105, 78153 le Chesnay, France. `malek.aichour@inria.fr`

[2] INRIA Rocquencourt - Domaine de Voluceau, BP 105, 78153 le Chesnay, France. `evelyne.lutton@inria.fr`

Summary. This work is a first step toward the design of a cooperative-coevolution GP for symbolic regression, which first output is a selective mutation operator for classical GP. Cooperative co-evolution techniques rely on the imitation of cooperative capabilities of natural populations and have been successfully applied in various domains to solve very complex optimization problems. It has been proved on several applications that the use of two fitness measures (local and global) within an evolving population allow to design more efficient optimization schemes. We currently investigate the use of a two-level fitness measurement for the design of operators, and present in this paper a selective mutation operator. Experimental analysis on a symbolic regression problem give evidence of the efficiency of this operator in comparison to classical subtree mutation.

1 Introduction

Within the core of bioinspired co-evolution techniques, various extensions of evolutionary algorithms have been used to efficiently tackle complex problems. Among them mono-population cooperative algorithms have been developed, based on problems decomposition. They can be considered as a link between pure cooperative agent-based approaches (like Ant Colony Optimizations [5]) and artificial Darwinism.

When it is possible to consider a problem as a collective learning task, the searched solution can be built from the whole set of individuals of an evolved population, and not only from its single best individual. The most famous techniques of this type are classifier systems [7], Parisian approach [3], and cooperative coevolution [19]. The Parisian approach has for example produced applications in text-retrieval[15, 16], in art and design[2], or even real-time applications (stereo-vision using the "flies algorithm" [17]).

The idea defended in this paper is that the design of new operators in variable length structure evolution like GP may stem from cooperative-coevolution schemes. Advanced GP operators may for example benefit from the idea of using two fitness functions within an evolutionary scheme. Considering locally optimized operators, attempts have been made on crossover, in order to decide the best crossing point based on local measurements [9, 18]. Local measurements can actually be considered as a "local fitness function" in the Parisian approach spirit. Here we consider locally optimized mutations in this way.

This work is a first step for the development of a Parisian approach to symbolic regression. The long term idea is to develop a set of simple test-function benchmarks for cooperative-coevolution optimization algorithms, to complete a first set of functions designed in [14].

The paper is organized as follows. Section 2 recalls the principles of Parisian evolution, then the use of a two level fitness evaluation for genetic operators design is considered (section 3). The selective mutation operator is presented in section 4, and tested on three instances of symbolic regression 5. Conclusions and future work are sketched in section 6.

2 The Parisian evolution cooperative co-evolution scheme

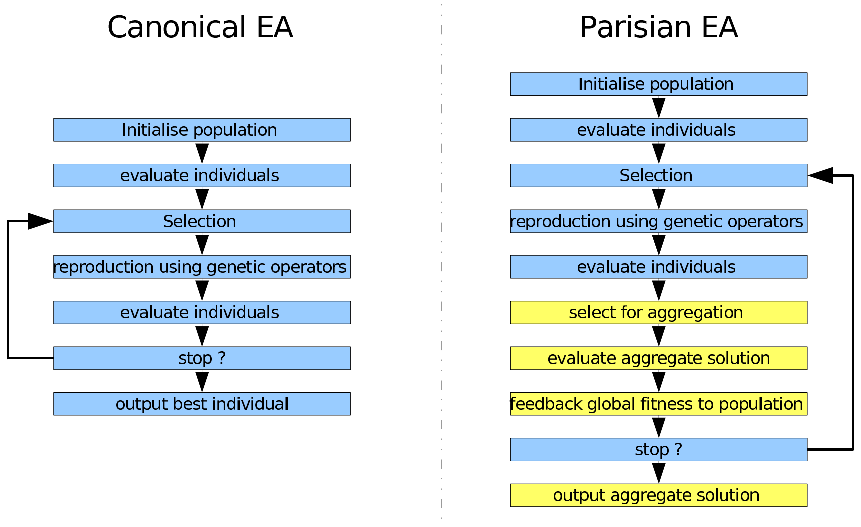


Fig. 1. Outline of a standard implementation of a Parisian EA. Fitness evaluation step is modified in order to consider local and global fitness.

This approach, originally proposed in [3], is based on a two-level representation of an optimization problem, in the sense that an individual of a Parisian population represents only a part of the problem solution. An aggregation of multiple individuals must be built in order to obtain a solution at hand. In this way, this is the whole population (or a major part of it) evolution that is favoured instead of the emergence of a single best individual, as in classical evolutionary schemes. The motivation is to make a more efficient use of the genetic search process, and reduce the computational expense. Successful applications of such a scheme usually rely on a lower cost evaluation of the partial solutions (i.e. the individuals of the population), while computing the full evaluation only once at each generation.

Figure 1 outlines the structure of a Parisian EA, for which many of the canonical aspects of EAs are retained. Additional characteristics are described below and aim at building a *society* of individuals that implicitly collaborate via the aggregation and global evaluation steps.

- **Encoding:** each individual of the population encodes a part of the solution.
- **Individuals aggregation:** at each generation a set of "best" local solutions is aggregated from the current best global solution.
- **Local fitness:** this fitness is calculated for each individual, and is an estimation of its potential contribution to the whole searched solution. Constraints on the problem can be introduced here, in order to prune useless computations for example.
- **Global fitness:** it is computed at each generation, and only on a potentially good aggregation of partial solutions. Longer computations can be performed here.
- **Evolutionary engine:** the complete population is evolved, and the selective pressure is intended to promote the emergence of better aggregate solutions. A scheme to aggregate local and global fitness values is usually required. Additionally a diversity preservation mechanism is necessary in order to maintain a complementary set of partial solutions in the current population.

The applicability of this approach is restricted to problems that can be split into homogeneous components, whose individual contribution to the solution can be evaluated. Each implementation is thus strongly application dependent. This approach actually relies on the following assumptions [6]:

1. A complete problem solution $X \in S$ can be decomposed into n components $x_i \in S'$, and there exists a mapping T: $S' \times S' \times ... \times S' \to S$.
2. There exists a merit function f_{loc} to evaluate each component.
3. The fitness landscape defined by f_{loc} and S' has sufficient structure to guide the search process toward the global optimum of the global fitness function f_{glob} in S.

A usual way to address item 3 is to implement a bonus distribution algorithm that distribute the value of f_{glob} computed at each generation on the partial solutions which participate to the global one. This mechanism has been designed to ensure a positive pressure on partial solutions that have been identified as good by the aggregation process.

3 Using a two-level evaluation process at the level of genetic operators

The previous Parisian cooperative co-evolution model is based on the use of two fitness functions, that in some way "collaborate." The design of such couple of partial/global fitness functions is of course extremely problem dependent. The challenge on this topic is certainly to produce a set of benchmark couples of functions on which various evolutionary engines may be compared. A first attempt has been made in [14]. We intend to continue in this direction by testing variable-length structures strategies and test-problems. GP representations seems actually well adapted to the philosophy of Parisian approaches which tend to gather simple structures into a global variable-size structure. Symbolic regression test-problems are currently tested.

In doing this, we first focused on the way the couple of local/global fitness functions can be used: as can be noticed the Parisian-style cooperation is implemented at the level of the selection/aggregation mechanism. This remark lead us to investigate if there are no additional ways to implement a two-level fitness cooperation within a GP engine.

The simplest way to do this is certainly within genetic operators, and indeed there already exist crossover operators that correspond to this idea.

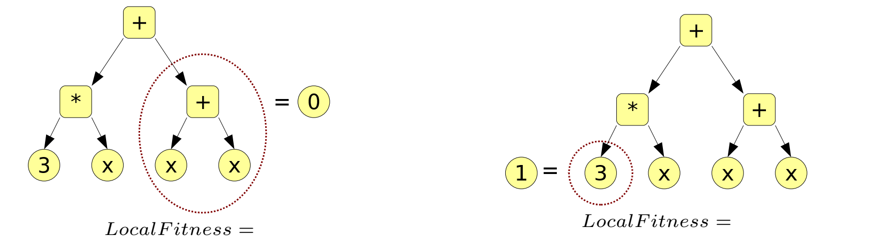
The standard way crossover is implemented is by exchanging genetic material via the choice of a node in each parent tree and exchange of subtrees [13], without taking into account the content of each subtree. This may prevent the emergence of structured solutions[4]. It is therefore argued that crossover behaves more like a macro-mutation operator [11] and may be not better than a mutation-only system [1].

Some content-aware crossover techniques have been proposed in the literature (selective crossover [9, 18]). The way the content of subtrees is taken into account can actually be related to a local fitness function. This raises the idea of extending this idea to mutation.

4 Selective Mutation

Standard *GP* mutation selects a mutation node randomly in the parent tree, and the corresponding sub-tree is replaced with a random one[12]. There are more sophisticated versions of GP mutation, like for example the *headless chicken crossover*[20] (i.e. a crossover with a temporary random tree), but to our knowledge none of them really takes into account the content of the subtrees.

The idea developed here is to identify the "worst" subtree of a candidate for mutation, and to replace it by a new random one. The "worst" subtree identification is made with the help of a "local" fitness function. In this way, a mutation point will allow to focus the search onto unused, useless or even deleterious areas of genomes, while (hopefully) minimizing the loss of "good"



$LocalFitness =$
$Fitness(3x + (x + x)) - Fitness(3x)$

$LocalFitness =$
$Fitness(3x + (x + x)) - Fitness(x + (x + x))$

Fig. 2. Selective mutation operator : local fitness evaluation

material at the individual level. We use the term *selective mutation* to refer to the selective crossover proposed in [9, 18], and to refer to an identified natural phenomenon in genetics[3].

The local evaluation is a measurement of the weakness of the sub-tree. In the present work it represents the contribution of the sub-tree in the context of its container-tree, but obviously, additional constraints on the genome can be embedded in the "local" fitness, in the same way as the classical Parisian approach.

The local fitness measurement, i.e. the contribution of a sub-tree is currently implemented the way Hammad and Ryan evaluate GP schemes [8]: the sub-tree to be evaluated is replaced by a neutral element (i.e. 1 for the multiplication, 0 for the addition, etc ...) and the resulting individual is evaluated. The difference between the fitness of the initial complete tree and of a tree in which a sub-tree has been replaced by a neutral element correspond to the evaluation of this sub-tree weakness, see figure 2. Iba and Garis [10] also suggested to measure the impact of sub-tree by treating each sub-tree as an independent program. However this evaluation does not necessarily indicates the real contribution of a sub-tree towards the main tree.

Once the local fitness can be evaluated on each subtree, the most simple strategy, which is tested below, i.e. an exhaustive search on all subtrees, yield the best candidate subtree to mutation. Of course other strategies can be imagined in order to spare computation time. A tournament selection based on a small sample of the possible subtrees may for example be programmed, to reduce the computational costs and preserve some randomness in the choice of the mutation node.

[3] A selective mutation is a change in a gene that is specifically selected, see `http://www.everythingbio.com/glos/definition.php?word=selective%20mutation`

5 The Experiments and Results

5.1 Symbolic Regression Problem

Symbolic regression problems involve generating functions which approximate a target function, f, given a set of data pairs, (x_i, y_i), such that $y_i = f(x_i)$. In real world applications, the target function is of course unknown, and the data pairs are obtained from empirical observations. We are therefore interested in inducing functions that provide an approximate fit of the data. For the purpose of testing algorithms on these tasks, it is common to derive fitness pairs from a known function and to evaluate each candidate solution according to how well it approximates the target. Since it is possible to derive many semantically equivalent functions to the target, the fitness function evaluates the error observed between the candidate and target values over a number of fitness points. There are many ways of doing this, for example summing the absolute differences between the observed and target values, taking the mean squared error, or taking the root of the sum of the squared differences. We should not expect a candidate solution to fit the target data perfectly so we should also specify a small tolerance margin of error, within which a fitness case is said to been correctly classified. The success criterion is therefore not a total error of *zero*, but an acceptable classification of all the fitness points. Since the aim of symbolic regression is the induction of a symbolic function, implementations include mathematical operators in the function set. The terminal set should include instances of each input variable. As an example, consider the target function $x^4 - x^3 + x^2 - x$. The function set for this function might be the four arithmetic operators $+, -, *, /$ whilst the terminal may consist of the single input variable x. The tree or individual (in postfix notation) : $xx * x - xxx * xx * x * - * -$ is an example solution to the fourth polynomial function $x^4 - x^3 + x^2 - x$.

Table 1. *GP* Parameters for Symbolic Regression Problems

Objectives	Find a program that produces the given value of the fourth quintic and sextic polynomial (respectively F_1,F_2 and F_3) as its output when given the value of the one independent variable, x, as input.
Stop criterion	Max generation or total error = 0.01.
Function set	$\{+, -, *, \%\}$ (protected division).
Terminal set	$\{x\}$
Fitness case	random values of x from the range $-1 \ldots + 1$.
Fitness	The mean, over the number of fitness cases, of the absolute value of the difference between the value returned by the program and the target function.
Initial pop	Created using ramped half-and-half.
Population size	100
Selection	Tournament selection.
Crossover	0.80
Mutation	0.20
Maximum depth	17 for F_1, 20 for F_2 and 25 for F_3
Maximum generation	200

5.2 Experiments

This section presents some results obtained with the proposed selective mutation operator, the GP which was coded in *Visual C++ 6.0*.

The three benchmark problems are symbolic regression of the fourth polynomial ($F_1 : x^4 - x^3 + x^2 - x$) [21], symbolic of the quintic polynomial ($F_2 : x^5 - 2x^4 + x^2$) [13] and symbolic regression of sextic polynomial ($F_3 : x^6 - 2x^3 + x$) [13].

The performance of our selective mutation GP *SMGP* is compared with a standard genetic program *SGP* with the traditional mutation operator. In both programs we used the crossover described in [12]. The default parameters for all runs are described in table 1.

For each target function, we use 50 independent runs with different seeds and we measure minimum and maximum, mean and median of fitness, number of generations and run time. As for these small instances both algorithms usually converge to a solution (fitness near 0), number of generations to convergence (error lower that a fixed threshold) and corresponding run time are more discriminant quantities.

The run time is measured in seconds and obtained on an *Intel Pentium Core Duo 2 Ghz - PC*. Table 2,3 and 4 shows the results.

After a series of tests we noticed that *SMGP* requires less CPU time than *SGP*, even if each mutation involves an exhaustive search on each subtree of a candidate. After calculating the statistical means of the number of generations to convergence, table 2, 3 and 4 show that *SGP* required significantly more generations than *SMGP*. Table 2 shows that in function F_1 *SMGP* has no failure run (line of fitness, all values are *zero*). The results show that with *SGP* the number of generations needed for convergence is highly unpredictable: for the function F_2, it managed to find the optimal solution in 30 generations at its best, and at its worst, 190 generations. And for the function F_1 it finds the optimal solution in 15 generations at its best, and in 90 generations at its worst.

Table 2. Results of function F_1

		Min	Max	Mean	Median	Std-Dev
	Fitness	0	$9.71445e^{-17}$	$9.12000e^{-03}$	$5.40000e^{-04}$	
SGP	#Generations	15	90	63.68000	50	21.8
	Run Time	$7.41000e^{-01}$	$2.98300e^{00}$	$1.71040e^{00}$	$1.89020e^{00}$	0.61
	Fitness	0	0	0	0	
SMGP	#Generations	9	80	28.34000	25	15.85
	Run Time	$2.16000e^{-01}$	$1.23000e^{00}$	$1.3820e^{00}$	$8.12100e^{-01}$	0.28

The test related in table 2, i.e. find the polynomial F_1 from 9 pairs (x_i, y_i) uniformly distributed in $[-1, +1]$, is the same as in [21], the result is however

Table 3. Results of function F_2

		Min	Max	Mean	Median	Std-Dev
SGP	Fitness	$4.61000e^{-03}$	$2.63640e^{00}$	$1.13500e^{-01}$	$1.62400e^{-01}$	
	#Generations	30	190	70.54000	60	33.2
	Run Time	$1.17102e^{00}$	$5.14982e^{00}$	$2.41758e^{00}$	$2.89300e^{00}$	1.05
SMGP	Fitness	0	8.71000^{-03}	9.40000^{-04}	0	
	#Generations	30	160	50.40000	50	20.35
	Run Time	$8.84100e^{-01}$	$4.60600e^{00}$	$2.18000e^{00}$	$2.36540e^{00}$	0.33

Table 4. Results of function F_3

		Min	Max	Mean	Median	Std-Dev
SGP	Fitness	$4.60000e^{-03}$	$2.63640e^{00}$	$1.13500e^{-01}$	$0.16240e^{00}$	
	#Generation	40	200	80.24000	90	26.89
	Run Time	$2.39300e^{00}$	$7.66450e^{00}$	$4.18610e^{00}$	$4.36201e^{00}$	1.16
SMGP	Fitness	0	$0.00960e^{00}$	$0.00013e^{00}$	0	
	#Generation	35	180	70.88000	70	30.04
	Run Time	$1.88200e^{00}$	$5.64300e^{00}$	$3.68440e^{00}$	$2.88600e^{00}$	0.85

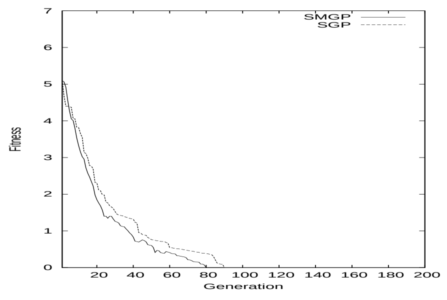
different. On average, for nearly 95% of runs SMGP gave the correct solution after up to 80 generations (most of the solutions were found before 25th generation). For this target function F_1, the maximum number of nodes in the tree was set to 17, a population of 100 individuals reached the exact solution in the generation 9 after 0.2160 seconds of execution. The mutation probability was set to 0.20. The best individual found is $xxxxxx*-*-*x-$ (in postfix notation) which is equivalent to the reference polynomial $x^4 - x^3 + x^2 - x$.

In the second example (table 3) we try to perform the symbolic regression of the data generated by the function F_2 in the interval $[-1, +1]$. The maximum number of nodes in the tree was set to 20, the population size was 100 and after 30 generations the exact solution was found by SMGP: $xx*xxx*xx***+xxx**x*xxxx***+-$ which is $x^5 - 2x^4 + x^2$.

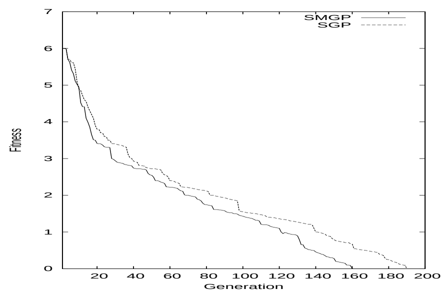
In the third example (table 4), the function F_3 was used to generate 20 pairs (x_i, y_i) uniformly distributed in $[-1, +1]$. The maximum number of nodes in the tree was set to 25. After 35 generations with SMGP, a population of 100 individuals evolved to the exact solution $xxxx**xx**xx*xx*+-*x+$ equivalent to $x^6 - 2x^3 + x$.

6 Conclusion and Future Work

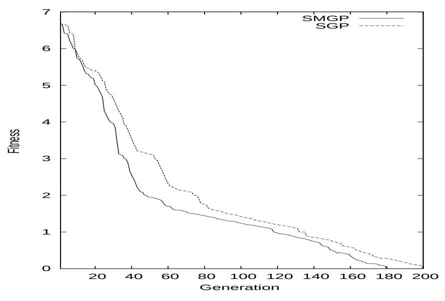
This work is a preliminary work, which allows to set the design of context-aware operators in a more general framework. A very simple scheme of selective mutation has been successfully tested on three symbolic regression



(a) F_1



(b) F_2



(c) F_3

Fig. 3. Fitness curves of *SGP and SMGP in functions F_1, F_2 and F_2.*

benchmark problems. The powerful idea is to focus the action of mutation to areas of the genome that are not correctly (or sufficiently) evolved, in order to favour the emergence of highly fit parts of solutions and maintain it in the population. For this purpose, the design and the use of an additional fitness function has been necessary. This local fitness function has characteristics similar with local fitness functions of Parisian approaches, in the way it favours the emergence of good partial solutions (=subtrees for GP).

Further work will consider the development of pure Parisian GP approaches on symbolic regression problems, as well as context-aware GP operators based on local fitness measurement. In the present work, the local fitness measurement is strongly correlated to the global fitness, we will consider other less correlated (but application-dependent) couples of local/global fitness functions, in order to induce more complex population dynamics.

References

1. P.J. Angeline. Subtree crossover: Building block engine or macro mutation ? In Genetic Programming 1997: Proceedings of the Second Annual Conference.

MorganKaufmann, July 1997.

2. J. Chapuis and E. Lutton. Artie-fract : Interactive evolution of fractals. In *4th International Conference on Generative Art*, Milano, Italy, Dec 12-14 2001.
3. P. Collet, E. Lutton, F. Raynal, and M. Schoenauer. Polar ifs + parisian Genetic Programming = efficient IFS inverse problem solving. *Genetic Programming and Evolvable Machines Journal*, 1(4):339–361, 2000. October.
4. P. D'haeseleer. Context preserving crossover in genetic programming. In Proc of the 1994 IEEE World Congress on Computational Intelligence, vol 1, pp 256261, Orlando, USA, 27-29 1994. IEEE Press.
5. M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. *New Ideas in Optimization*, pp 11–32, 1999. D. Corne, M. Dorigo and F. Glover, editors, McGraw-Hill.
6. E. Dunn, G. Olague and E. Lutton. "Parisian Camera Placement for Vision Metrology" In Pattern Recognition Letters, Vol. 27, No. 11, August, pp. 1209-1219, 2006.
7. L. Bull and T. C. Fogarty. Co-evolving communicating classifier systems for tracking. pp 522–527, Innsbruck, Austria, April 1993. Springer-Verlag, Wien.
8. M. Hammad, C. Ryan. A new approach to evaluate GP schema in context In Genetic an Evolutionnary Computation Conference (GECCO 2005) workshop program (Washington, D.C., USA, 25-29 June 2005), F. Rothlanlf at AL., Ads., ACM Press PP. 378. 381
9. S. Hengpraprohm and P. Chongstitvatana. "Selective Crossover in Genetic Programming". citeseer.ist.psu.edu/536164.html
10. Iba, H., and Garis, H., Extending Genetic Programming with Recombinative Gidance, angeline, P. and Kinnear, K., editors, Advanced in Genetic Programming vol 2, MIT Press, 1996.
11. R.E. Keller, W. Banzhaf, P. Nordin and F. D. Francone "Genetic Programming An Introduction" Morgan Kauffman, 1998.
12. Koza, J. R., Genetic Programming : On the Programming of Computers by Natural selection. MIT Press, Cambridge, MA. 1992.
13. J.R. Koza. Genetic Programming II: Automatic Discovery of reutilisable programs. MIT Press, Cambridge Massachusetts, May 1994.
14. G. Ochoa, E. Lutton, E. Burke. "Cooperative Royal Road: avoiding hitchhiking". In Evolution Artificielle 2007. Tours, France.
15. Y. Landrin-Schweitzer, P. Collet, and E. Lutton. Interactive gp for data retrieval in medical databases. In *EUROGP'03*. LNCS, Springer Verlag, 2003.
16. Y. Landrin-Schweitzer, P. Collet, E. Lutton, and T. Prost. Introducing lateral thinking in search engines with interactiveevolutionary algorithms. In *SAC 2003, Special Track COMPAHEC, 2003.* Melbourne, Florida, U.S.A.
17. J. Louchet, M. Guyon, M.-J. Lesot, and A. Boumaza. Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. *Pattern Recognition Letters*, 2002. No. 23 pp. 335-345, Elsevier Science B.V.
18. C. K. Mohan. Selective Crossover: Towards Fitter Offspring. Symposium on Applied Computing (SAC'98), Atlanta. 1998.
19. M. A. Potter and K. A. De Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1):1–29, MIT Press, 2000.
20. R. Poli, and N. McPhee, Exact GP Schema Theory for Headless Chiken Crossover and Subtree Mutation. 2001.
21. W. B. Langdon Quick Intro to simple-gp.c, University College London, 1994.