Genetic Programming without Fitness

Andrea G. B. Tettamanzi
Università degli Studi di Milano, Dipartimento di Scienze dell'Informazione
Via Comelico 39/41, I-20135 Milano, Italy.
E-mail: tettaman@dsi.unimi.it
URL: http://www.dsi.unimi.it/∼ tettaman/index.html

ABSTRACT

This paper provides a short, informal illustration of a selection scheme based on the key idea of competition, particularly suited for genetic programming, which provides a way to do without the explicit definition of a fitness function.

In many tasks, competition between two individuals on one problem instance chosen according to some probability can be a valid alternative to defining an appropriate fitness function that includes *a priori* knowledge of the problem, which requires insights on the problem along with some mathematical skills.

1 Introduction

The workings of evolutionary algorithms (EAs) (Michalewicz 1992, Bäck and Schwefel 1993), including genetic algorithms (Holland 1975, Goldberg 1989), evolution strategies (Schwefel 1981) and evolutionary programming (Fogel, Owens and Walsh 1966), critically rely on the existence of a fitness function, which mirrors the objective function of the problem to be solved.

For some problems a meaningful fitness function is trivial to come up with, especially when the problem is already mathematically formulated, as is the case with classical problems from operations research. However, for most problems approached by GP, like symbolic regression, robot control etc., translating a desired behavior into an objective function is a formidable task, for the following reasons:

- A desired behavior is expressed by means of fuzzy statements like "avoid obstacles" or "fit a curve";
- There is an infinity of equally meaningful ways to measure distance between an observed behavior and desired behavior, let alone the fact that often there ere infinitely many distinct behaviors that satisfy a given specification (i.e. the family of L_n norms, like the integral of the difference, mean square error and so on).

Even when an objective function is given, appropriate definition of the fitness function is critical to the effectiveness of the evolutionary algorithm.

After all, however, there must be a way of doing without having to define a fitness function: indeed the measure of fitness in Nature is the probability for a genotype of reproducing. Competition between two individuals to obtain some scarce resource plays a central role in natural selection, which appears to be a local, distributed and asynchronous process. Examples in Nature of this kind of competition could be a fight to secure enough food, the duels between bucks to gain the right to mate with a given female and so on. Based on this observation, the idea of competitive selection is introduced.

The claim is that it is enough to have an ordinal criterion which tells which of two individuals is better in a determined circumstance, not in general.

2 Competitive selection

Competitive selection (Tettamanzi 1994) is a form of stochastic binary tournament selection, where the notion of fitness is never used, at least directly.

Tournament selection, attributed to a never published work by Wetzel and studied for the first time in Brindle (1981), extracts a small number of individuals from the current population with uniform probability and without replacement and has them play a "tournament". In the deterministic case, the tournament is won by the fittest individual; the tournament can be probabilistic, the probability of an individual winning it being proportional to its fitness. Tournament selection has enjoyed large popularity and has produced much literature (see Baker (1985), Baker (1987), Mühlenbein (1989) and Goldberg (1990) among others).

Competitive selection depends on a stochastic function which, given two competing individuals, selects the winner according to a probability. Formally, if we denote with Γ the set of all possible genotypes, **winner**: $\Gamma \times \Gamma \to \Gamma$ must obey the following properties:

1. $P[\mathbf{winner}(\gamma, \kappa) = \gamma] + P[\mathbf{winner}(\gamma, \kappa) = \kappa] = 1$;
2. $P[\mathbf{winner}(\gamma, \kappa) = \gamma] = P[\mathbf{winner}(\kappa, \gamma) = \gamma]$.

2.1 Definition of winner

The **winner** function provides a simple device to evaluate relative individual performance in a very direct way. Two examples lying at the extremes of a vast range of possible situations should illustrate how to proceed in defining the **winner** function:

1. Symbolic regression (Arnone, Dell'Orto and Tettamanzi 1994): we want to fit as closely as possible a given "unknown" function (what is unknown is its formula, but we assume we are able to evaluate it in any point). A random point in the function's domain of definition is generated according to a given probability distribution function and is then fed as input to both competitors; the individual that, evaluated in that point, gets closer to the actual value of the function wins the competition.

2. Controller optimization (Tettamanzi 1995): a random initial state is generated according to a given probability distribution function, then the behavior of both competitors is emulated on two parallel simulations of the system to be controlled starting from the same state. A competitor loses as soon as it violates any constraint of the problem; a competitor wins if one of the following three cases happens:

   (a) its opponent loses;

   (b) the simulation comes to an end (because of time/out or at random) and the controller is in a target state, while its opponent is not;

   (c) the simulation comes to an end and the controller is "closer" to the goal than its opponent is.

   In all other cases we say that the competition is tied and the winner is chosen by flipping a fair coin. Selection pressure can be adjusted by the use of multiple competition matches.

2.2 Implementation

Letting n be the size of a population, we can use the **winner** function to build a general selection scheme as follows:

for $i = 1$ **to** n **do**
  begin
    $\gamma :=$ *old_pop*$[\mathcal{U}(1, n)]$;
    $\kappa :=$ *old_pop*$[\mathcal{U}(1, n)]$;
    new_pop$[i] := \mathbf{winner}(\gamma, \kappa)$;
  end;

where $\mathcal{U}(1, n)$ denotes picking an integer from 1 to n with uniform probability.

This is exactly the algorithm for binary tournament selection. The only difference is that here a stochastic **winner** operator is used to determine which individual will reproduce instead of a criterion based on fitness.

This scheme can be easily parallelized: for example in a massively parallel architecture each Processing Element (PE), holding, say, γ, could pick a κ at random among its immediate topological neighbors, then store $\mathbf{winner}(\gamma, \kappa)$ on board in the place of γ.

Let us denote $n_x(\gamma)$ the number of individuals in population x sharing the same genotype $\gamma \in \Gamma$. The expected number of occurrences of γ in the new population y obtained from population x by applying the competitive selection scheme given above is

$$E[n_y(\gamma)] = \frac{2n_x(\gamma)}{n} \sum_{\kappa \in \Gamma} n_x(\kappa) P[\mathbf{winner}(\gamma, \kappa) = \gamma].$$

3 Observations

In both the symbolic regression and the controller optimization cases the **winner** probability depends on a probability distribution function used to generate a random evaluation point or initial state. That's a bit of this approach that remains "arbitrary" to some extent. This distribution could be chosen such that it gives more importance to regions of the problem domain that are known or thought to occur more often in practice. In absence of information of that kind, a zero-information distribution might be a sensible choice.

With its sampling of the whole problem domain, competitive selection overcomes one difficulty of selection in GP, namely the question whether the fitness cases are (1) a sufficient and (2) representative set with respect to the "real" problem in practice. First, their choice by the programmer contains a dangerous amount of arbitrariness; second, evolving programs that perform well on those cases doesn't give any guarantees on their performance on other cases, even if contained in the convex hull of fitness cases, because in principle we don't know whether the evolved programs work as (possibly non-linear) interpolators or they treat each fitness case as separate.

Experiments performed by the author have shown that an evolutionary algorithm making use of this competitive selection scheme performs at its best when the crossover rate is rather low, i.e. less than 20%. This can be explained by the fact that some time has to be allowed to compensate for statistical deviations in the outcomes of competitions, before the genetic material of an individual is exchanged or modified.

4 Relationships with fitness

Of course, as the reader has probably realized, fitness is still there, lurking behind the scene. What competitive selection eliminates is the need of explicitly expressing it, which is a difficult and dangerous process.

Indeed, theorems have been proven in Tettamanzi (1994) which show the equivalence between competitive selection and random binary tournament selection.

5 Conclusions

In a number of tasks that are typical for the application of GP, competition between two individuals on one randomly chosen problem instance can be a valid alternative to defining an appropriate fitness function that includes *a priori* knowledge of the problem.

Besides just making the design of a GP application simpler, competitive selection guarantees a higher degree of generality for the solutions, as it samples the whole problem domain and not just a limited number of fitness cases.

Acknowledgments

The author would like to thank Prof. Gianni Degli Antoni for his patient support and incitement and Prof. Vincenzo Capasso for the useful discussions and criticism.

Bibliography

Arnone, S., Dell'Orto, M. and Tettamanzi, A. 1994. Toward a fuzzy government of genetic populations. In *Proceedings of the 6th IEEE Conference on Tools with Artificial Intelligence (TAI'94)*. Los Alamitos, CA: IEEE Computer Society Press. Pages 585–591.

Baker, J. 1985. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette (editor), *Proceedings of the First International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum Associates. Pages 101–111.

Baker, J. 1987. Reducing bias and inefficiency in the selection algorithm. In John J. Grefenstette (editor), *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum Associated. Pages 14–21.

Brindle, A. 1981. Genetic agorithms for function optimization. Technical Report TR81-2, Department of Computer Science, University of Alberta, Edmonton.

Bäck, T. and Schwefel, H.-P. 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, Volume 1, Number 1 (Spring). Pages 1–23.

Fogel, L. J., Owens, A. J. and Walsh, M. J. 1966. *Artificial Intelligence through Simulated Evolution*. New York: John Wiley & Sons.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. 1990. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, Volume 4. Pages 445–460.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

Mühlenbein, H. 1989. Parallel genetic algorithms, population genetics and combinatorial optimization. In J. D. Schaffer (editor), *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann. Pages 416–421.

Michalewicz, Z. 1992. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer-Verlag.

Schwefel, H.-P. 1981. *Numerical optimization of computer models*. Chichester, New York: Wiley.

Tettamanzi, A. 1994. A distributed model for selection in evolutionary algorithms. Technical Report 110-94, Dip. di Scienze dell'Informazione – Università degli Studi di Milano, Milano, Italy.

Tettamanzi, A. 1995. An evolutionary algorithm for fuzzy controller synthesis and optimization. Proceedings of the 1995 IEEE International Conference on System, Man and Cybernetics. Pages 4021–4026.