# Variable Size Population for Dynamic Optimization with Genetic Programming

Leonardo Vanneschi
Department of Informatics, Systems and Communication (D.I.S.Co.)
University of Milano-Bicocca, Milan, Italy
vanneschi@disco.unimib.it

Giuseppe Cuccu
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Lugano, Switzerland.
giuse@idsia.ch

## ABSTRACT

A new model of Genetic Programming with variable size population is presented in this paper and applied to the reconstruction of target functions in dynamic environments (i.e. problems where target functions change with time). The suitability of this model is tested on a set of benchmarks based on some well known symbolic regression problems. Experimental results confirm that our variable size population model finds solutions of the same quality as the ones found by standard Genetic Programming, but with a smaller amount of computational effort.

## Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming

## General Terms

Algorithms, Performance

## Keywords

Genetic Programming, Variable Size Population, Dynamic Optimization

## 1. INTRODUCTION

Many real-world problems are anchored in dynamic environments, where some element of the problem domain, typically the target, changes with time. In the last few years, many contributions have appeared which studied dynamic optimization environments and developped new evolutionary frameworks for solving them. Nonetheless, for the majority of those approaches the problem objective is finding the extrema (maxima or minima) of a target function that changes with time. On the other hand, very few contributions have appeared to date studying the ability of Genetic Programming (GP) to reconstruct target functions on dynamic optimization environments. In this paper we hypothesize that variable size population GP outperforms standard (fixed size) GP on dynamic optimization problems. This idea is not new in evolutionary computation; for instance, it has been applied to PSO in [2]. However, it has never been applied to GP before. We propose a variable size population GP model called *DynPopGP*, inspired by the one presented in [4] and we compare its performance with the one of standard GP on a set of new symbolic regression-like dynamic optimization benchmarks.


*GECCO'09,* July 8–12, 2009, Montréal Québec, Canada.
ACM 978-1-60558-325-9/09/07.

## 2. THE PROPOSED ALGORITHM

*DynPopGP* can be summarized by the pseudo-code in Figure 1, where we consider minimization problems (i.e. small fitness values are better than large ones). This algorithm uses the following functions: (i) $update\_pop\_size(x)$, that adds $|x|$ individuals to the population if $x$ is positive and suppresses $|x|$ individuals if $x$ is negative; and (ii) $\Delta_{pop}()$, that returns the number of individuals that have to be to be added or suppressed from the population when neither the old best fitness value nor the new one approximate the optimal fitness value in a satisfaisable way. The $\Delta_{pop}()$ function performs the following calculation: $\Delta_{pop}() = pivot \cdot strength \cdot best\_fit\_contribution() \cdot pop\_size\_contribution()$, where: (iii) *pivot* is a variable whose value is $-1$ if the best fitness in the population at the current generation is better then the one at the previous generation and $+1$ otherwise (in practice, the value of *pivot* determines if individuals have to be added or suppressed); (iv) *strength* is a variable that determines how strong populations inflate and deflate have to be at each step; (v) the $best\_fit\_contribution()$ function, that determines the contribution given to the $\Delta_{pop}()$ by the best fitness value reached (it is defined by the pseudo-code in Figure 2); and (vi) the $pop\_size\_contribution()$ function, that is analogous to the $best\_fit\_contribution()$ function, except that it determines the contribution to the $\Delta_{pop}()$ given by the current population size.

## 3. TEST PROBLEMS

It would make no sense to use moving peaks benchmarks as the ones presented for instance in [1] in this work. In fact, in those kinds of benchmark, extrema are moved by changing some additive or multiplicative constants to a (otherwise not changing) target function. If one uses GP with linear scaling (introduced in [3]), the moving peaks problem reduces to a static GP problem, given that linear scaling allows to reconstruct the shape of the target functions, offering a method to automatically determine additive and multiplicative constants. For this reason, we define a new set of benchmark problems that can be used to test GP ability to reconstruct target functions in dynamic environments. Maintaining the same terminology as in [3], we have considered test functions $F_{12}$, $F_{13}$, $F_{14}$, $F_{15}$ and $F_{16}$ and we have used them to build dynamic test problems in which the importance of the modification of the target function can be tuned. Using these test functions, we have built three benchmarks for dynamic optimization that we have called BENCH1, BENCH2 and BENCH3. The target function at each generation is calculated by the algorithm in Figure 3, where given a test function $F_i$, with $12 \leq i \leq 15$ $succ(F_i) = F_{i+1}$ and $succ(F_{16}) = F_{12}$.

```
begin
    Generate a population of N random individuals;
    best = best individual in the population;
    old_got_trg = false;
    for g := 1 to maxgen do
        new_got_trg = (fitness(best) ≤ trg_fit);
        if (not new_got_trg)
          then
            elitism (i.e. copy of the best);
            selection;
            reproduction / crossover;
            mutation;
            best = best individual in the new population;
            new_got_trg = (fitness(best) ≤ trg_fit);
            if (old_got_trg)
                then
                  // The old best had reached the target, while
                  // the new best has not reached it:
                  // the target function has surely changed.
                  // Set the population size to the initial size
                  update_pop_size(N - current_pop_size);
                else
                  // Neither the new best, nor the old best
                  // have reached the target: update the
                  // population size using the Δ_pop function
                  update_pop_size(Δ_pop());
            endif
          else
              if (not old_got_trg)
              then
                  // The new best has reached the target, while
                  // the old best had not reached it. This means
                  // that the target has been found now.
                  // I have to spend as few computational effort
                  // as possible until the target function changes
                  // (or the process terminates).
                  // I set the population size to a prefixed
                  // "stand-by" value
                  update_pop_size(stand_by_size - current_pop_size);
              endif
          endif
          old_got_trg = new_got_trg;
    endfor
end
```

**Figure 1: Pseudo-code for the *DynPopGP* algorithm.**

```
best_fit_contribution() ::
    if ( fitness(best) ≤ trg_fit) then return min_coeff;
    elsif (fitness(best) ≥ max_fit) then return max_coeff;
    else return
        max_coeff − min_coeff · (fitness(best) − trg_fit) / (max_fit − trg_fit) + min_coeff
    endif
```

**Figure 2: Pseudo-code for the *best_fit_contribution* function.**

```
begin
    Define a set of test functions F = {f_1, f_2, ..., f_n}
    for g := 1 to maxgen do
        For each fitness case (x, y), the target value is: Σ_{i=1}^{n} f_i(x, y)
        if (g mod period = 0) then
            ∀1 ≤ i ≤ n : f_i := succ(f_i)
        endif
    endfor
end
```

**Figure 3: Pseudo-code for target calculation in benchmark problems BENCH1, BENCH2 and BENCH3 The difference between these benchmark is in the size of set $F$: $n = 2$ for BENCH1; $n = 3$ for BENCH2 and $n = 4$ for BENCH3.**

# 4. EXPERIMENTAL RESULTS

In Figure 4 we report the results obtained executing 100 independent runs against generations for standard GP (*stdGP*) and *DynPopGP* for BENCH3. This figure clearly shows that: (a) the two GP models find solutions of similar qualities at corresponding generations; (b) the effort spent by *DynPopGP* is smaller than the one spent by *stdGP*; (c) solutions are found by *DynPopGP* with less computational effort than by *stdGP*; and (d) the population size of *DynPopGP* is always smaller than the one of *stdGP*. We also remark that, as expected, the population size of *DynPopGP* tends to grow at each *period* generations (generation number multiple of 20), because of the modification in the target function. Results for BENCH1 and BENCH2, not shown here for lack of space, are qualitatively analogous to the ones for BENCH3.

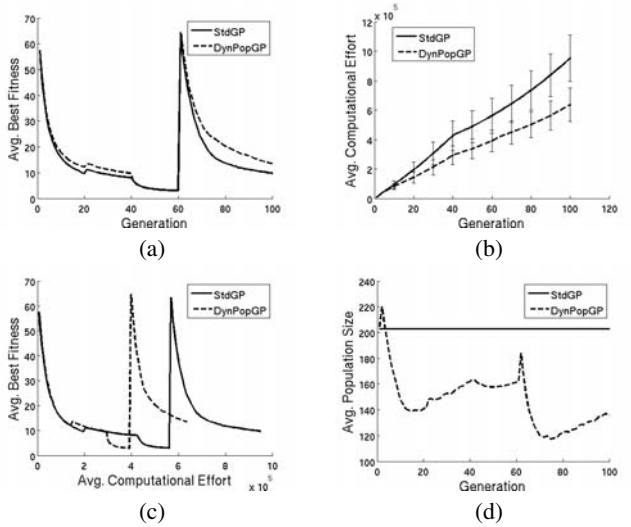


**Figure 4: Experimental results obtained using *stdGP* and *DynPopGP* for BENCH3. (a): Average best fitness against generations; (b): Computational effort against generations; (c): Average best fitness against computational effort; (d): Population size against generations.**

# 5. REFERENCES


[1] J. Branke. Evolutionary approaches to dynamic optimization problems – introduction and recent trends. In J. Branke, editor, *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 2–4, 2003.

[2] C. Fernandes, V. Ramos, and A. Rosa. Varying the population size of artificial foraging swarms on time varying landscapes. In *International Conference on Artificial Neural Networks: Biological Inspirations*, volume 3696 of *LNCS*, pages 311–316. Springer, 2005.

[3] M. Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In C. Ryan *et al.*, editor, *Genetic Programming, Proceedings of the 6th European Conference, EuroGP 2003*, volume 2610 of *LNCS*, pages 71–83, Essex, 2003. Springer, Berlin, Heidelberg, New York.

[4] M. Tomassini, L. Vanneschi, J. Cuendet, and F. Fernández. A new technique for dynamic size populations in genetic programming. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'04)*, pages 486–493, Portland, Oregon, USA, 2004. IEEE Press, Piscataway, NJ.