Title and Abstract

Title: Comparing the Lamarckian and Baldwinian Approaches in Memetic Optimization

Abstract: Memetic optimization (MO) combines local and global search in optimization in non-monotonic, 'rugged' search spaces. In particular, MO extends genetic optimization, where global search is implemented via the crossover operator and local search is performed as random mutation. MO uses more elaborate techniques to implement local search and to combine it with its global counterpart. This study is set to compare two ways of memetic optimization: one motivated by the Baldwinian, while the other by the Lamarckian theory of evolution.

Both the Baldwinian and Lamarckian theory suggest that behaviors of individuals are not only passed on to offspring by crossover and mutation, but also through lifetime learning. In Baldwin approach, learned behaviors affect the mapping of genotypes to phenotypes, which ultimately results in changes to the fitness landscape. In Lamarck approach, not only will the learned behaviors affect the fitness landscape in the same way as the former does, but they will also be passed on to offspring through phenotypes. Whilst the biological plausibility of these perspectives is questionable, they offer a valuable structure for constructing memetic optimization algorithms.

Before exploring Lamarckian and Baldwinian approaches, a baseline framework where offspring can only inherit the characteristics of the parent through crossover and mutation (i.e., genetic optimization) is considered as a global optimization problem. Based on the baseline framework, we develop various implementations of the Lamarckian and Baldwinian approaches exploring several local search procedures to study the potential contributions of the studied approaches. Our experiments will be performed on the CEC-BC-2017 test functions for optimization.

Keywords: Baldwinian evolution; Lamarckian evolution; Memetic optimization; Fitness landscape; Learning; CEC-BC-2017.

Genetic and Memetic Algorithm

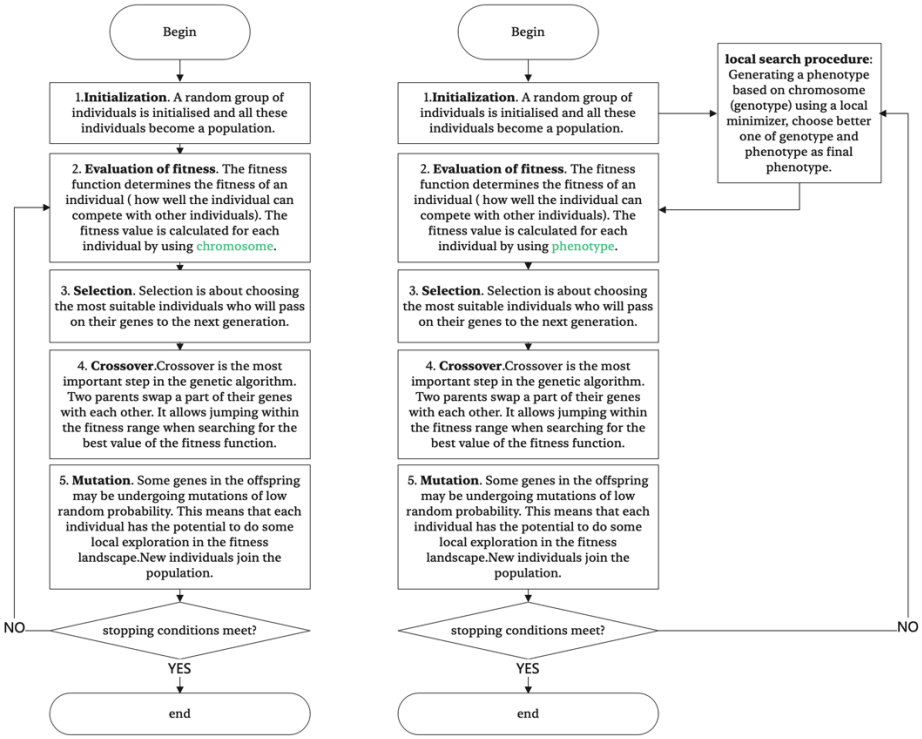
Genetic algorithm

Genetic algorithms originated from the Darwinian theory of evolution, by selection of superior individuals in the population to produce offspring, combined with genetic variation, with the expectation of producing individuals better adapted to an environment. The main three operators: selection, crossover and mutation of GA enable it to be applied to a wide range of global optimization problems, where each individual's chromosome(genotype) is a solution to a fitness (objective) function. Figure 1(left part) presents the flow of a simple genetic algorithm. The evolution starts from a population with randomly generated individuals. GA measures the fitness value of each individual and selects the most adapted individuals, then creates new offspring by crossover and mutation with a certain probability. New individuals will join the population and participate in the next iteration. After continuous iteration, GA evolves towards better solutions, but it is not guaranteed to find the global optima every time, depending on factors such as the number of iterations and the complexity of the objective function.

Memetic Algorithm

MO extends genetic optimization by introducing local search procedures. MO not only offers the selection, crossover, and mutation operators available in GA, but more importantly, MO leverages the concept of inheritance to provide a local optimizer, which can be considered as special kind of genetic search in a subspace, (Radcliffe & Surry, 1994). Crossover allows jumping within the fitness range when searching for the best value of the fitness function while mutation provides individuals with the potential to do some local exploration in the fitness landscape. MO further reinforces local search on the basis of mutation and crossover.

In genetic algorithms, each individual has its own chromosome(genotype), which is a solution to an objective function. MO will generate a phenotype in the vicinity of the fitness landscape where the genotype is located, with the help of a local optimizer. The better of genotype and phenotype will be involved in the later operations. As a general rule, the improvement between genotype and phenotype can be interpreted as the result of a lifelong learning effort which might be passed on to future generations as an inheritance. Figure 1(right part) presents the flow of a simple memetic algorithm.



Stopping Criteria

Stopping Criteria for GA and MO are generally organized as follows:

A. Max number of iterations is reached
B. A satisfactory fitness value of objective function has been found
C. The similarity of populations is less than a pre-defined threshold for certain number of iterations

Methods

Steady-state Genetic Algorithm

Steady-state genetic algorithm (SSGA) is implemented as a baseline framework in order to demonstrate the progress and improvements which Lamarckian and Baldwinian approaches provide. SSGA maintains a stable population size by generating only one new offspring based on the best individual in the population, while discarding the least adapted individual. The individual holding the lowest position in the fitness landscape is defined as the best individual, as our goal is to find the global minimum. In contrast, the individuals who are least adapted to the environment have the highest position. For all individuals in SSGA, genotypes and phenotypes are vectors of equal dimensions and equal values at corresponding positions in every test function of CEC-BC-2017. Fitness value of an individual is computed based on its phenotype. The genotypes of the primordial populations are randomly created with every gene located within the domain of a test function.

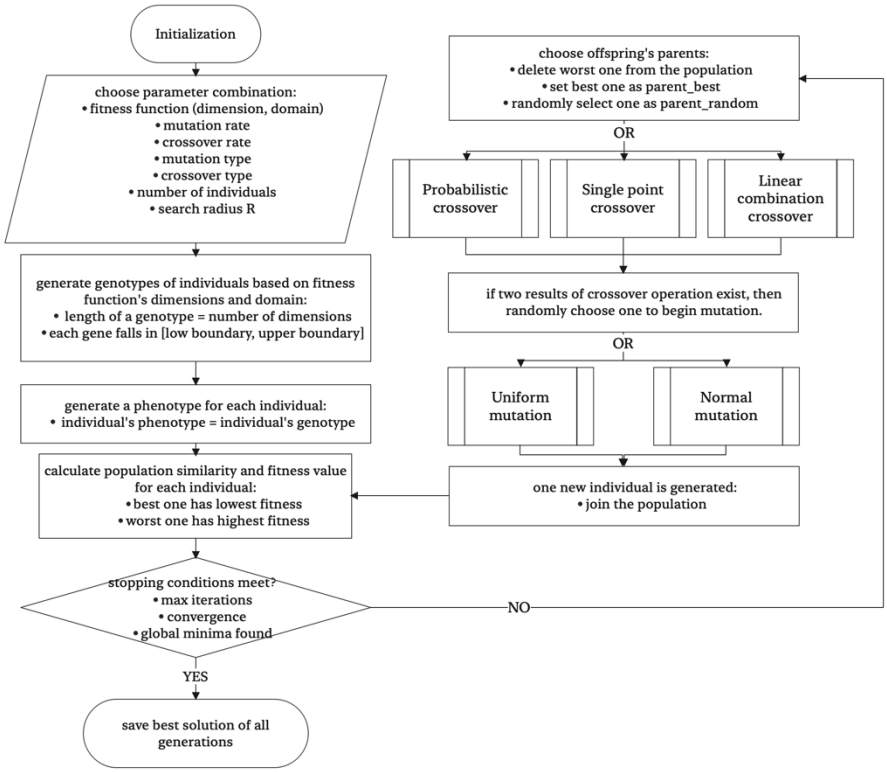


Figure 1Flowchart of SSGA

Figure 1 illustrates the flowchart of the SSGA algorithm. SSGA primarily consists of the following steps: (1) initialization of the population based on the parameter combination; (2) evaluation of the fitness value and similarity for the population; (3) selection of parents eligible to produce offspring and extinction of ineligible individuals; (4) crossover operation; (5) mutation operation; (6) insertion of new individuals into the population; (7) execution of next iteration.

The following **Pseudocode1** shows pseudo-code of SSGA. Parameter search radius R is designed to control the range of variation in mutation operation. The standard deviation of the normal mutation is equal to the difference between the upper and lower bounds of the fitness function multiplied by R. The range of the uniform mutation is equal to 6 times R times the difference between the upper and lower bounds of the fitness function, 3 times on each side of the x-axis. Each genotype or phenotype is a solution to selected fitness function. Every gene in genotype is a value in one dimension of fitness function and within its domain.

Pseudocode1

Input: parameter combination

1. Max iterations Max_{iter}
2. fitness function F , each dimension of F is F_i, domain of F is $[F_{low},F_{upper}]$
3. mutation rate γ
4. crossover rate δ
5. mutation type M_t
6. crossover type C_t
7. number of individuals β
8. search radius R

Output: best solution X

// **(1) initialization of the population based on the parameter combination**
Randomly generate β feasible genotypes g of individuals based on F
Create phenotypes p of individuals where $p_i = g_i$
Save all the individuals in the population Pop
Set iteration $iter = 1$
// **(2) evaluation**
// **(2.1) evaluation of the fitness value**
For $k = 1$ to β do:
Calculate F_p^k fitness value for an individual k using its phenotype p
Sort all the F_p^k in an ascending order and change order of individuals' location in Pop accordingly
Best individual has smallest fitness value $F_p^{k=1}$ while worst individual has the largest fitness value $F_p^{k=\beta}$
Save best solution of this generation $iter_{best}$ in iteration list $Iter\ list$
// **(2.2) evaluation of similarity for the population**
For $k = 2$ to β do:
calculate Euclidean similarity S^k between best individual and individual k
Sum all the S^k and save it as $iter_{similarity}$ in similarity list $Similarity\ list$
// **(3) selection of parents eligible to produce offspring and extinction of ineligible individuals**
Set best individual as best parent B_{parent}
Delete the worst individual from Pop
Randomly choose another parent as R_{parent} from Pop
// **(4) crossover operation**
If crossover type C_t = "Probabilistic Crossover":
do Probabilistic Crossover (crossover rate δ, B_{parent}, R_{parent})
If crossover type C_t = "Single point Crossover":
do Single point Crossover (crossover rate δ, B_{parent}, R_{parent})
If crossover type C_t = "Linear combination Crossover":
do Linear combination Crossover (crossover rate δ, B_{parent}, R_{parent})
// **(5) mutation operation**
If mutation type M_t = "Uniform mutation":
do Uniform mutation (mutation rate γ,search radius R, crossover result)
If mutation type M_t = "Normal mutation":
do Normal mutation (mutation rate γ,search radius R, crossover result)
// **(6) insertion of new individuals into the population**
Calculate the fitness value for this new individual F_p^{new} based on its phenotype
Insert F_p^{new} in fitness list and insert this new individual in Pop without breaking the ascending order
Save best solution of this generation $iter_{best}$ in iteration list $Iter\ list$
Repeat (2.2)
$iter = iter + 1$
// **(7) execution of next iteration**
If stopping conditions do not meet:
Repeat (3) (4) (5) (6) (7)
else:
output X minimum of best solution in all iterations

Crossover and Mutation operators of SSGA

Three kinds of crossover operators are implemented in SSGA in this paper.

A. Single point Crossover

Given crossover rate δ, first generate a random probability σ, if $\sigma < \delta$, then begin the procedure of crossover operation, otherwise randomly choose one of the two parents as the result of crossover.

Figure 2 illustrates the single point crossover process, with the best parent in green and the random parent in red. B_1 denotes the first gene of the best parent, while B_N denotes the last one. N is equivalent to the dimension of the fitness function. R_1 and R_N are also represented in the same way. The process of single point crossover is to select a random crossover point and then swap the best parent and the random parent for all the genes following that point.

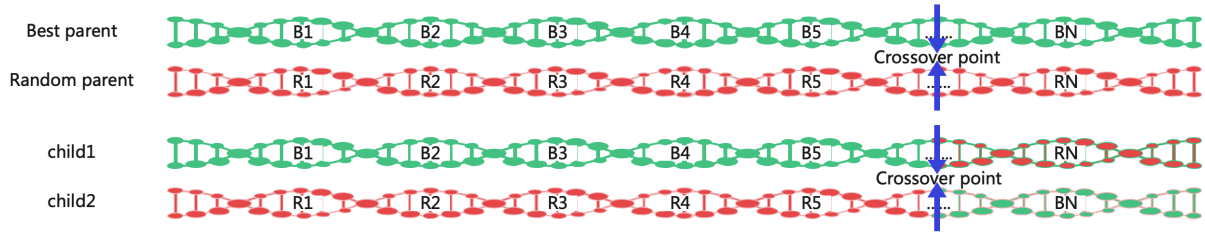


Figure 2 Single point crossover

B. Probability Crossover

Given crossover rate δ, best parent, random parent, and number of fitness function's dimensions N, for each gene of child, the probability of getting the best parent's gene is the same as crossover rate δ while the probability of inheriting a gene from a random parent is 1-δ.

$$P(C_i = B_i) = \delta$$
$$P(C_i = R_i) = 1 - \delta$$

Where $i \in 1,2,3,\ldots N$.

C. Linear combination Crossover

Given crossover rate δ, best parent, random parent, and number of fitness function's dimensions N, the child always inherits the characteristics of two parents.

$$P(C_i) = B_i * \delta + R_i * (1 - \delta) \text{ where } i \in 1,2,3,\ldots N.$$

Two kinds of mutation operators are implemented as following:

A. Uniform mutation

The probability density function of the continuous uniform distribution is:

$$f(x) = \begin{cases} \dfrac{1}{b-a} & \text{for } a \le x \le b \\ 0 & \text{for } x < a \text{ or } x > b \end{cases}$$

The lower boundary and upper boundary for variable x is a and b. More specifically, $-a = b = 3 * R * (F_{upper} - F_{low})$ where F_{upper} and F_{low} represent the upper and lower bounds of the fitness function respectively.

Given mutation rate γ and result of crossover, for each value in the result of crossover, generate a random probability σ, if $\sigma < \delta$, then plus a variable x generated by uniform distribution, otherwise remain the same.

B. Normal mutation

The probability density function of the continuous normal distribution is:

$$f(x) = \frac{1}{\sigma\sqrt{2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

x = value of the variable

$\mu =$ the mean = 0

$\sigma =$ the standard deviation = $R * \left(F_{upper} - F_{low}\right)$

Given mutation rate γ and result of crossover, for each value in the result of crossover, generate a random probability σ, if $\sigma < \delta$, then plus a variable x generated by normal distribution, otherwise remain the same.