Tuning Generalized Mutation Operators for Evolutionary Algorithms

Alexis Robbes

Tuning Generalized Mutation Operators for Evolutionary Algorithms

Alexis Robbes

Université de Technologie de Troyes, LIST3N
alexis.robbes@utt.fr

Keywords : *black box optimization, evolutionary algorithm, auto-configuration*

1 Introduction

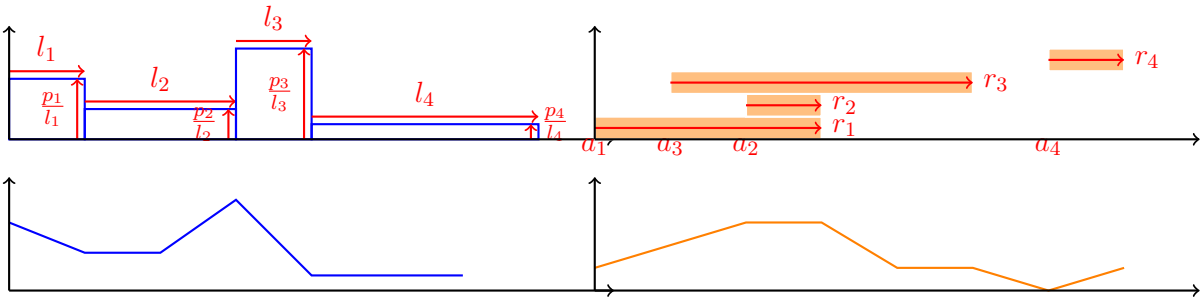
A large set of optimization problems can be presented as a binary problem where any solution is encoded as a bit string of n bits and the fitness function is a pseudo-Boolean function. This convenient encoding helps the development of general operators for Evolutionary Algorithms (EA) as bit mutation operators based on probabilistic law such as the binomial law or uniform law or normal law. These bit mutation operators' principle is to determine the number of bits k and then flip k random bits from the bit string. The most recommended bit mutation operator was for a period to always flip 1 bit ($flip_1$) or the standard bit mutation (binomial distribution) with a mutation rate of $\frac{1}{n}$. However several works have shown an interest in considering other operator distributions. In [2] a survey of the recent progress in the theory of EA is proposed. It is presented a list of research papers on the theoretical convergence rate of $(1 + \lambda)$ EAs with various mutation operators. In [4], the authors proposed a new mutation operator named Fast mutation for the (1+1) EA. It is shown that using a standard bit mutation operator is not optimal for the $(1 + 1)$ EA in particular when there is a jump function. One improvement was to flatten the operator distribution creating a "fat tail". Indeed, getting out of a local optimum may require an important mutation. Since then, the Fast mutation has demonstrated its efficiency in other contexts [1, 2] The authors of [3] numerically computed optimal unary unbiased mutation operators for the $(1 + \lambda)$ Evolutionary Algorithm (EAs) optimizing OneMax.

1.1 Generalized bit mutation

To represent any possible distribution, the most naïve way is to directly assign a value to $P(X = k)$, the probability of flipping k bits, for each in $\{1, \ldots, n\}$, where n is the bit string size. We name this operator Generalized Bit Mutation GBM. However, configuring GBM is theoretically difficult. In fact, it is a question of optimizing a vector of n parameters respecting a norm of 1. To ease the configuration step and don't lose too much generality, we suggest two trade-off strategies:

1. GBM buckets: consist to define a fixed number of connected buckets. The strategy is to determine for each bucket $b \in B$ its length l_b and its corresponding probability p_b. The inner distribution of each bucket is uniform. Figure 1a illustrates the two sets of parameters, $L = (l_b)_{b \in B}$ and $P = (p_b)_{b \in B}$, and the resulting distribution.

2. GBM segments: consist to define a fixed number of segments. The strategy is to determine for each segment $s \in S$ the first number of bits represented by a_s and its length r_s. The inner distribution of each segment is uniform and when segments overlap, the probabilities are summed. Figure 1b illustrates the two sets of parameters, $A = (a_s)_{s \in S}$ and $R = (r_s)_{s \in S}$, and the resulting distribution.



(a) GBM buckets parameters (top figure) and respecting distribution (bottom figure).

(b) GBM segments parameters (top figure) and respecting distribution (bottom figure).

2 Quality measure

Comparing operators should be done using a proper criterion. To avoid unnecessary computation time if we assume that the optimal fitness is known, we fix a budget for the number of evaluations and stop the EA when either the budget is reached or the optimal fitness is reached. As we want to configure operators on problems from various difficulties, we define a criterion based on the required number of fitness evaluations compared to the evaluation budget and on the best fitness found compared to the optimal fitness. The quality measure is then described by Equation 1 and should be minimized.

$$\textbf{quality} = \frac{\textbf{evaluations}}{evaluationbudget} + \frac{optimalfitness - \textbf{fitness}}{\textbf{fitness}} \tag{1}$$

This metric is easily interpretable as shown in Table 1.

quality	**evaluations**	**fitness**
≤ 1	quality $\cdot$ evaluation budget	optimal fitness
> 1	evaluation budget	$\frac{optimalfitness}{quality}$

TAB. 1: interpretation of the quality measure

References

[1] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. Fast mutation in crossover-based algorithms. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 1268–1276, 2020.

[2] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. Fast mutation in crossover-based algorithms. *Algorithmica*, 84(6):1724–1761, 2022.

[3] Maxim Buzdalov and Carola Doerr. Optimal static mutation strength distributions for the (1+ λ) evolutionary algorithm on onemax. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 660–668, 2021.

[4] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Proceedings of the genetic and evolutionary computation conference*, pages 777–784, 2017.