Fast Algorithm for Fair Comparison of Genetic Algorithms

Chia-Sheng Chen
Taiwan Evolutionary Intelligence Laboratory
Department of Computer Science and Information Engineering
National Taiwan University
b03902003@ntu.edu.tw

Hung-Wei Hsu
Taiwan Evolutionary Intelligence Laboratory
Department of Electrical Engineering
National Taiwan University
b02901023@ntu.edu.tw

Tian-Li Yu
Taiwan Evolutionary Intelligence Laboratory
Department of Electrical Engineering
National Taiwan University
tianliyu@ntu.edu.tw

ABSTRACT

Since numerous genetic algorithms (GAs) are developed every year, GA researchers need a fast algorithm to fairly compare their performances. In this paper, we formalized the performance metric and listed three algorithms to find the right population size for performance comparing in terms of the Number of Fitness Evaluations (NFE). Instead of finding the population n_{mNFE} producing minimum NFE (mNFE), we took the methodology of finding n^* which would converge to an arbitrary notion of success with a desired probability p^*. Among all three algorithms, the first, the most commonly used bisection method, was proved to be biased and without generality. The second is an unbiased modification of the first with trade-off of more function evaluations. The third, called Greedy Approach Regarding Locality (GARL), is our recommendation, empirically outperforming the second one by an exponential factor. We also analyzed the time complexity of the second and third algorithms, providing the upper bound for an average case. This work could be viewed as a general efficiency-comparing framework to almost all GAs except for parameterless schemes.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**;

KEYWORDS

Genetic algorithms, parameter tuning, performance measures, population

ACM Reference format:
Chia-Sheng Chen, Hung-Wei Hsu, and Tian-Li Yu. 2018. Fast Algorithm for Fair Comparison of Genetic Algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference, Kyoto, Japan, July 15–19, 2018 (GECCO '18),* 8 pages.
https://doi.org/10.1145/3205455.3205511

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205511

1 INTRODUCTION

Since a number of genetic algorithms (GAs) are being proposed every year, an interesting question is raised: how should the researchers compare the efficiency of different GAs? As in difficult, real-world problems, the fitness evaluation usually requires the most computational resources [12], so a proper way to measure is to count the number of fitness evaluations (NFE) when the population converges to optimum, or more generally, to an arbitrary notion of success defined for each problem.

But it is still hard to compare multiple GAs given just the lone indicator, NFE. NFE is widely defined to be infinity if the GA eventually fails to solve the problem, meaning that the population size being 0 results in an infinite NFE. On the other side, GAs must do at least one fitness evaluation for each chromosome of the first generation, so that a population approaching infinite size also leads to an infinite NFE. In between there shall exist some population size with desirable NFE.

A common practice is to find the minimum NFE (mNFE) that the GA needs to consume before success, and let different GAs compare to each other based on their mNFE. This approach is theoretically sound but not especially empirically practical. Since the number of generations to solve the problem is not fixed, a number of procedures must be run per population size to approach the expected value. To find n_{mNFE}, the population size that yields mNFE, the algorithm must statistically distinguish between the expected values of two adjacent population sizes, while there are, say, N candidates for n_{mNFE}. Let the distinguishment takes $O(m)$ trials each, the total effort is of an intolerable complexity of $O(mN)$.

GA researchers have come up with designs to deal with this problem. One way is to sweep the population size through a reasonable range with a predefined increment, so that the range is narrowed around the least-NFE population size. Then sweep the new range with a smaller step in order to find n_{mNFE} [9]. Another way is to find the minimal population n^* able to achieve successful convergence a certain times out of a certain independent runs and take the NFE consumed as the measurement. The above procedure was usually carried out by a bisection algorithm [12, 13, 16, 17, 19], but, as shown later in Section 3, this bisection algorithm is problematic.

Therefore, in this paper, we illustrate a more general scheme to this problem, addressing the effect of population size on the NFE of a GA. Observe that, although other parameters of GA, such as crossover rate or mutation rate, might have a more influential effect on the NFE, this work still focuses on the effect of different

population size, because contemporary GAs, especially the estimation of distribution algorithms (EDAs) [14], or called probabilistic model-building genetic algorithms (PMBGAs) [11], including hierarchical Bayesian optimization algorithm (hBOA) [12], linkage tree GA (LTGA) [1], and DSMGA-II [9], tend to eliminate all other parameters except for the population size. Regardless of the population-size-to-NFE curve, almost all GAs can be compared under this scheme. The only exceptions are the parameterless schemes, such as the parameter-less genetic algorithm [7], the parameterless hierarchical BOA [15], the Parameter-less Population Pyramid (P3) [6], and the parameterless DSMGA-II [3]. Since these further remove the population size as the parameter, these are beyond the scope of this work.

For the sake of efficiency, the work presented in this paper is to find n^* instead of n_{mNFE}. The major contributions of our work are: (1) clarifying the current measurement, NFE, of the efficiency of GA, (2) formalizing the metric, and (3) proposing a cost-efficient algorithm for finding the most suitable population size as the basis for fair comparison of various GAs. To the best of our knowledge, this is the first work which formally addresses this problem.

The rest of this paper is organized as follows: in Section 2, we describe the problem in a more formal manner. Afterward, a naïve and intuitive bisection method is proposed in Section 3. We also state that the naïve algorithm is biased and not generally applicable, which is a significant flaw in practice. To solve the biasedness, in Section 4, we offer an unbiased modification to the original one. We then also introduce a more cost-effective algorithm called the Greedy Approach Regarding Locality (GARL) in Section 5. Later in Section 6, we demonstrate the comparison of algorithms from that of Section 4 and 5 (§6.1), as well as show the applicability (§6.2) and some empirical suggestions (§6.3). The conclusion comes at last in Section 7.

2 PROBLEM FORMULATION

Let $p_{\mathcal{A};n}$ be the probability of successful convergence of the genetic algorithm $\mathcal{A}$ with population size n. The goal is to find a set of population size $(n_1, \ldots, n_k)$ such that for a list of GAs $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$, the successful convergence rate $p_{\mathcal{A}_1;n_1} = p_{\mathcal{A}_2;n_2} = \cdots = p_{\mathcal{A}_k;n_k} =$ a given desired rate p^* to a fitness function $f(\cdot)$.

Since this process might involve many diverse algorithms, to the best generalizability we do not assume any model for the population-size-to-success-rate (n-to-p) curve. Nonetheless, there still exist some properties for the problem regardless of the algorithm $\mathcal{A}$, and let p_n be $p_{\mathcal{A};n}$ defined previously with no concern of the algorithm $\mathcal{A}$ (or when the algorithm $\mathcal{A}$ is specified and not causing any ambiguity):

(1) $\forall a < b, p_a < p_b$. Namely, the more population, the higher chance the GA to achieve an arbitrary notion of success.
(2) $p_0 = 0, \lim_{n\to\infty} p_n = 1$. The algorithm always fails with the population size being 0, whereas always succeeds with the population size approaching infinity.
(3) $\Omega \sim \mathcal{A}(n) \in \{-1, +1\}, \forall \mathcal{A}, n \in \mathbb{Z}_0^+$. The outcome space contains only *positive* (solve the problem, converge to optimum, etc.) and *negative* regardless of the algorithm $\mathcal{A}$.

Because of the preceding property (1), given a desired successful convergence (*convergence* in short) rate $p^* \in [0, 1)$, the goal is defined to find $n^* \stackrel{\text{def}}{=} \text{argmin}_n\{|p_n - p^*|, n \in \{0, 1, \ldots, N\}\}$ for each evolutionary algorithm, where N is a hyperparameter in this process for the maximum possible population size. Since the exact number requires lots of computing effort, a small proportion of error ϵ is tolerable. That is, answers in range $[n^* - \epsilon N, n^* + \epsilon N]$ are accepted. Whilst it takes infinite trials to output the answer with 100% confidence, we allow a residual of δ.

To put it even more briefly, the process aims to find a population n^* given a 5-tuple $(\mathcal{A}, p^*, N, \epsilon, \delta)$ as the hyperparameter.

3 NAÏVE BISECTION ALGORITHM

3.1 Algorithm

Since convergence rate monotonically increases against the population size n. An intuitive way is to find the desired population size n^* by the bisection method [16]. More detailed, the method is to do multiple trials at a point and determines the next trial point accordingly. Say, population sizes of two GAs are tuned to a comparable standard: each GA has a chance of 50% to converge 10-time-in-a-row. If so, the desired population size should infer a convergence rate $p^* \approx 0.93$, given by $0.93^{10} \approx 0.5$.

The naïve bisection, in this case, works as follows: 10 trials are tossed at a certain n. If all of them succeed, then with the population size n it is expected to have the convergence rate larger than 0.93. So the process searches toward the smaller direction of the population size domain, and vice versa. Or, extensively, m outcomes are sampled, then the process decides the newer range by which there are $\leq k$ successes. Due to the uncertainty of Bernoulli trials, the average over results of multiple bisections will be outputted instead of one single bisection. Which is, provided that the outcome of the previous process was modeled by a random variable $\mathcal{H}$, the inference of the process is $\mathbb{E}[\hat{\mathcal{H}}]$. But, in fact, the above naïve algorithm, also recorded in algorithm 1, is biased.

Algorithm 1 Mean of Bisection

```
repeat
    l ← 0
    r ← N
    while r − l > max(εN, 1) do
        c ← ⌊(l + r)/2⌋
        Sample m outcomes from A(c)
        if ≤ k successes then l ← c else r ← c
    end while
    Append(results, ⌊(l + r)/2⌋)
until one is happy
n* ← ReducedMean(results)
return P(n*) ≈ p*
```

3.2 Problems of Naïve Method

In this subsection, we tried to show that the naïve bisection is problematic from two aspects. Firstly, the above criteria, i.e. 10-out-of-10 moves to the left, otherwise right, does not suggest an expected value close to 0.93. We will formulate the situation and sketch up

some equations in Section 3.2.1 to show that it is a lot more complicated. And secondly, it is not generally applicable for all $p^* \in [0, 1)$. We will reveal that in Section 3.2.2.

3.2.1 Problem 1 – Complication. For the bisection algorithm, initially, there is a range for all possible candidates, $[0, N)$. The procedure then tosses m times from $\mathcal{A}(\lfloor N/2 \rfloor)$. If there are less than or equal to k successes, which implicitly means $\hat{p}_{\lfloor N/2 \rfloor} \leq p^*$, then the range is narrowed down to $[\lfloor N/2 \rfloor, N)$, and vice versa, to $[0, \lfloor N/2 \rfloor)$. The same routine proceeds until there is only one element in the range, serving as the output to the problem. Evidently, the value of the output is not deterministic because of the uncertainty of coin tossing. Let $\mu(m, k) : \Omega \to \mathbb{Z}$ be the random variable of the output of the procedure with m tosses per round and k successes as the threshold. The target of this section is to disprove that the convergence rate with the population size $\mathbb{E}[\mu(10, 9)]$ approximates to 0.93, or, more generally, to show that $\mathbb{E}[\mu(m, k)]$ is impractical to compute given arbitrary m, k.

The preceding expected value is shown in a recursive form. Let $E_{m,k}(l, r)$ be the expected value with m as the number of tosses, k as the threshold, and $[l, r)$ as the range of all possible candidates. In this instance, $\mathbb{E}[\mu(m, k)] = E_{m,k}(0, N)$.

$$E_{m,k}(l,r) = \begin{cases} p_l, & \text{if } l+1 = r \\ xE_{m,k}\left(\left\lfloor \frac{l+r}{2} \right\rfloor, r\right) + (1-x)E_{m,k}\left(l, \left\lfloor \frac{l+r}{2} \right\rfloor\right), & \text{else} \end{cases} \tag{1}$$

where

$$x = I_{1-p_{\lfloor (l+r)/2 \rfloor}}(m-k, k+1). \tag{2}$$

The term $I_{1-p}(\cdot)$ in Eq. 2 is the cumulative distribution function (CDF) of the binomial distribution, representing the probability of having less than or equal to k successes with p as the success probability in each trial.

Since the desired expected value $\mathbb{E}[\mu(m, k)]$ is obtained from $E_{m,k}(0, N)$, the troublesomeness of the procedure is shown undoubtedly. It is not only painful to calculate all of the recursion, but also, more crucially, *dependent* upon the underlying population-size-to-convergence-rate curve of the problem by the term $p_{\lfloor (l+r)/2 \rfloor}$ in Eq. 2, requiring all convergence rate from p_0 to p_N to be known. It therefore violates the main target of the procedure, which is to find $\operatorname{argmin}_n\{|p^* - p_n|\}$.

3.2.2 Problem 2 – Not Generally Applicable. Just opposite from what was mentioned in Section 3.2.1: to find p given m, k is not a concern in this problem. The actual key problem is to find m, k given p, so that the procedure is realizable by tossing m samples per round, and decide the direction by which there are more than k successes. But, it is not generally possible. Suppose that there exists an arbitrary $m \in \mathbb{Z}^+$. Since there are only finite ordered pairs (i, j) which $0 < i \leq m, 0 \leq j < i$, the set of its mapping $\{p_{\mathbb{E}[\mu(i,j)]} \mid 0 < i \leq m, 0 \leq j < i\}$ is also finite. But $[0, 1)$ is an uncountably infinite set [2], so there always exists a $p \in [0, 1)$ such that toss per round greater than m is needed. As long as m is unreasonably large, it is not practically applicable.

4 UNBIASED MODIFICATION

4.1 Algorithm

An unbiased modification to the bisection algorithm was proposed as follows: for every decision, the algorithm keeps experimenting on same population, say n', until there is a high enough confidence indicating that n^* is on either side of n'. Since there should be $\lceil \lg N \rceil$ decisions, to let the total residual be in δ, the residual for a single decision should be in $\delta' = 2\left(1 - (1-\delta)^{\frac{1}{\lceil \lg N \rceil}}\right)$. The algorithm is demonstrated in algorithm 2, named as Precise Bisection (PB).

Algorithm 2 Precise Bisection

```
l ← 0
r ← N
while r − l > max(εN, 1) do
    m ← ⌊(l + r)/2⌋
    repeat
        Sample from A(m)
    until P(p̂_m < p*) < δ′ or > 1 − δ′
    if p̂_m ≥ p* then l ← m else r ← m
end while
return ⌊(l + r)/2⌋
```

4.2 Time Complexity

The time complexity of precise bisection is $\Theta\left(\lg \epsilon^{-1} \cdot f(n)\right)$, where $\lg \epsilon^{-1}$ is the raw time complexity for bisection, $f(n)$ representing the trials needed for fulfilling the inequation $\mathbb{P}(p_n < p^*) < \delta'$ (for simplicity, hereinafter, what works works for $\mathbb{P}(p_n < p^*) > 1 - \delta'$ as well). Given single-bounded Hoeffding inequality [8] regarding Bernoulli random variables,

$$\mathbb{P}(H(f(n)) \leq (p - \varepsilon)f(n)) = \delta \leq \exp(-2\varepsilon^2 f(n)), \tag{3}$$

the new residual δ' can be bounded by

$$\begin{aligned} \mathbb{P}(p_n \leq p^*) = \delta' &\leq \exp\left(-2(p^* - p_n)^2 f(n)\right) \\ f(n) &\geq \frac{\ln \delta'}{-2(p^* - p_n)^2}. \end{aligned} \tag{4}$$

If the equality in Eq. 4 is established, the number of $f(n)$ is bounded by the second best candidate $n^\dagger$, which is the last decision to make before termination.

$$\begin{aligned} f(n) &= O\left(-(p^* - p_{n^\dagger})^{-2} \ln \delta'\right) \\ &= O\left(-(p^* - p_{n^\dagger})^{-2}(-\lg \lg N + \lg \delta)\right). \end{aligned} \tag{5}$$

Hence, let Δ be $p^* - p_{n^\dagger}$, the unbiased modification — Precise Bisection (PB) is of the order shown in Eq. 6.

$$\Theta\left(\lg \epsilon^{-1} \cdot f(n)\right) = O\left(-\lg \epsilon \Delta^{-2}(\lg \lg N - \lg \delta)\right). \tag{6}$$

5 GREEDY APPROACH REGARDING LOCALITY (GARL)

We proposed a novel technique to find the right population size n_{mNFE}. It requires a n-to-p curve $\mathcal{P}$, acting as the prior belief of the

convergence rate for all population size, and a weight w, indicating the confidence of the belief, as the additional hyperparameters. Since a solution ranging in $[n^* - \epsilon N, n^* + \epsilon N]$ would be considered correct, not every candidate from 0 to N is needed here. Instead, one of averagely every $2\epsilon N$ candidates could represent the rest, what we call these representator *bar* henceforth. We therefore reformulated the problem into picking the best bar b^* from a total number of $\left\lfloor \frac{1}{2\epsilon} \right\rfloor + 1$ bars B. To put it more accurately, what interests us is a range $[\cdot, \cdot]$ formed by a (best) bar with one of its neighbors such that n^* has high probability to lie in the range. The inference of the algorithm is therefore the midpoint of the range.

$$B_i \stackrel{\text{def}}{=} \text{Round}((i-1) \cdot \epsilon N). \tag{7}$$

$$\tau_i \stackrel{\text{def}}{=} \text{the number of samples on } B_i. \tag{8}$$

$$\tau_i^{\{+,-\}} \stackrel{\text{def}}{=} \text{the number of pos./neg. outcome from } \mathcal{A}(B_i). \tag{9}$$

The methodology here is to sample from the closest possible point $b^\times$ suggested by the prior curve. If $b^\times$ is optimal, then the action is also optimal to terminate. But if other bar looks better, it switches instantly. Finally, it sticks on a best bar b^* until termination.

To determine the best point, the simplest way is to greedily pick the bar whose expected convergence rate τ^+/τ is closest to p^*. But, owing to (1) the continuity and (2) the increaseness of the underlying n-to-p curve, the convergence rate of an arbitrary bar b is highly correlated with its left-hand-side (LHS) and right-hand-side (RHS) neighbors $LHS(b)$ and $RHS(b)$. Moreover, $p_{LHS(b)} < p_b < p_{RHS(b)}$, and $\nexists b' \neq b$ such that $p_{LHS(b)} < p_{b'} < p_{RHS(b)}$. So, in this paper, we came up with a concept to better decide which bar to sample. To leverage the good of *locality*, the expected convergence rate $\tilde{p}_i$ is not defined as τ_i^+/τ_i, but as $\sum_{j=i-1}^{i+1} \tau_j^+ / \sum_{j=i-1}^{i+1} \tau_j$, taking samples from neighbors into consideration as well. This can also be viewed as applying the smoothing method to the convergence rates with window size = 1.

Algorithm 3 Greedy Approach Regarding Locality

Precondition: $\mathcal{P}, N, w$
  for $i \in [1 : |B|]$ **do** ▷ Preprocessing, prior installation.
    $\tau_i \leftarrow w$
    $\tau_i^+ \leftarrow w \cdot \mathcal{P}_{B_i}$
    $\tau_i^- \leftarrow w \cdot (1 - \mathcal{P}_{B_i})$
  end for
  repeat
    $\tilde{p}_i := \sum_{j=i-1}^{i+1} \tau_j^+ / \sum_{j=i-1}^{i+1} \tau_j$
    $\hat{b} \leftarrow \text{argmin}_i |\tilde{p}_i - p^*|$ ▷ Greedily find the closest point.
    Sample 1 outcome from $\mathcal{A}(\hat{b})$
    if result $= +1$ **then** $\tau_{\hat{b}}^+ \leftarrow \tau_{\hat{b}}^+ + 1$ **else** $\tau_{\hat{b}}^- \leftarrow \tau_{\hat{b}}^- + 1$
    $\tau_{\hat{b}} \leftarrow \tau_{\hat{b}} + 1$
  until $\pi_1(\textsc{Term}(\hat{b}))$ = True ▷ See Sec. 5.1 for $\textsc{Term}(\cdot)$.
  $b^* \leftarrow \hat{b}$
  return $\pi_2(\textsc{Term}(b^*))$ ▷ $\pi_{\{1,2\}}$ for 1st, 2nd projection.

The prior belief curve $\mathcal{P}$ is installed with a weight w representing how reliable the prior is. The larger the weight, the harder it is to change the inferred $\hat{b}$, and also the faster it terminates if the prior correctly represents the underlying model. But, on the other hand, a falsely suggested prior would produce a wrong outcome.

5.1 Termination Condition

A possible condition for termination is to ensure that p^* is in between two adjacent bar b_1 and b_2 with $1-\delta$ probability. When the condition is met, the process output $\lfloor (b_1+b_2)/2 \rfloor$ so that the output is guaranteed to be in $[n^* - \epsilon N, n^* + \epsilon N]$.

Supposed there are two random variables representing the distribution of underlying expected convergence rate p_1, p_2 for b_1 and b_2, where $p_1 < p_2$ for the sake of simplicity, named X_1, X_2. The aim is to find the probability of p^* being in between X_1 and X_2.

Formally speaking, if the goal is to keep the percentage of residual in δ, then the terminated formula would be

$$\mathbb{P}(X_1 \leq p^* \leq X_2) \geq 1 - \delta. \tag{10}$$

Both X_1 and X_2 follow beta distribution, whose pdf is

$$f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1} du} = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \tag{11}$$

where $B(\cdot)$ is Beta function. More precisely, given that for X_i, τ_i times of trials are tossed and τ_i^+ positive rewards are received, $i \in \{1, 2\}$. The probability for X_i having the underlying convergence rate p_i is $f(p_i; \tau_i^+ + 1, \tau_i^- + 1)$.

Therefore, the terminate condition is

$$\begin{aligned}
&\mathbb{P}(X_1 \leq p^* \leq X_2) \geq 1 - \delta \\
\Longrightarrow\ &\mathbb{P}(p^* \leq X_1) + \mathbb{P}(p^* \geq X_2) \leq \delta \\
\Longrightarrow\ &1 - I_{p^*}(\tau_2^+ + 1, \tau_2^- + 1) + I_{p^*}(\tau_1^+ + 1, \tau_1^- + 1) \leq \delta \\
\Longrightarrow\ &I_{p^*}(\tau_2^+ + 1, \tau_2^- + 1) - I_{p^*}(\tau_1^+ + 1, \tau_1^- + 1) \geq 1 - \delta.
\end{aligned} \tag{12}$$

For a bar b, the function $\textsc{Term}(b)$ could be defined as whether p^* falls in either $[p_{LHS(b)}, p_b]$ or $[p_b, p_{RHS(b)}]$ with a high probability. The function is described in algorithm 4. It is worthwhile to mention that the function returns a 2-tuple in order to distinguish the left-range from the right-range, since both would reply a "True" if passing the terminate condition.

Algorithm 4 Terminate Condition

function $\textsc{Term}(b)$
    if $\mathbb{P}(p_{LHS(b)} < p^* < p_b) > 1 - \delta$ **then**
      return (True, $\lfloor (LHS(b) + b)/2 \rfloor$)
    else if $\mathbb{P}(p_b < p^* < p_{RHS(b)}) > 1 - \delta$ **then**
      return (True, $\lfloor (b + RHS(b))/2 \rfloor$)
    else
      return (False, $\cdot$)
    end if
  end function

5.2 ϵ-GARL

Although barely happens, the proposed algorithm GARL actually comprises an efficiency blind spot: as shown in Figure 1, the process could already passed termination condition $\mathbb{P}(p_1 < p^* < p_2) >$

$1-\delta$ inasmuch as $\hat{p_1}$ being somewhat closer to p_1, but the algorithm will keep sampling from b_2 because $d(p^*, \hat{p_2}) < d(p^*, \hat{p_1})$, which helps little toward termination. In fact, the greediness will lead to an arbitrary long process in extreme condition. But if b_1 is further sampled, by central limit theorem $\hat{p_1}$ will converge to p_1, and hence the termination.

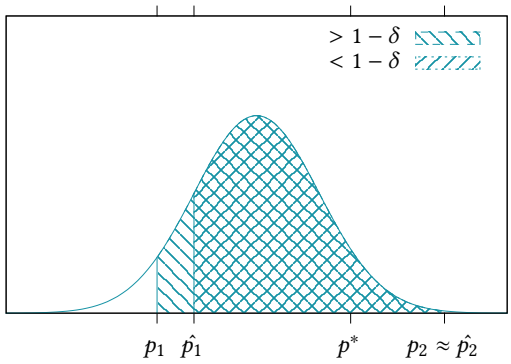

Figure 1: The probability of $p^* \in [p_1, p_2]$

A proper solution to this is, intuitively, to sample on the neighbors of b^* once in a while in order to let their estimation being close to the real value. Therefore we applied the idea from ϵ-greedy [18] to this issue. Instead of always letting $\hat{b} = \text{argmin}_i |\tilde{p_i} - p^*|$, ϵ-GARL samples from LHS and RHS neighbors with non-zero probability α, where $\alpha = 0.1$ is a decent choice.

$$\hat{b} = \begin{cases} LHS(\text{argmin}_i |\tilde{p_i} - p^*|), & \text{with probability of } \alpha \\ RHS(\text{argmin}_i |\tilde{p_i} - p^*|), & \text{with probability of } \alpha \\ \text{argmin}_i |\tilde{p_i} - p^*|, & \text{with probability of } 1-2\alpha \end{cases} \tag{13}$$

5.3 Time Complexity

The time complexity of GARL can be bounded by the sum of an exploration part $f(\cdot)$ and an exploitation part $g(\cdot)$.

The exploration part $f(\cdot)$ could be understood as follows: supposing that $b^\times \overset{\text{def}}{=} \text{argmin}_b |\mathcal{P}_b - p^*|$, being the closest-to-$p^*$ bar according to prior $\mathcal{P}$, is not the final bar b^*. The algorithm therefore should convince itself that b^* is better than $b^\times$ through a series of trials. Since $b^\times$ is the best candidate at first, GARL would first sample from $b^\times$ until it seems incorrect. And by the locality had been mentioned previously, its neighbor $LHS(b^\times)$ or $RHS(b^\times)$ would seem to be the best then. With no loss of generality, let $b^\times > b^*$. The process could be viewed as at some point, $|p^* - p_{LHS(b^\times)}| < |p^* - p_{b^\times}|$, then the algorithm starts to experiment on $LHS(b^\times)$, which we called a *hand-over* henceforward. Accordingly, the exploration phase consists of hand-overs at $b^\times, LHS(b^\times), \ldots, RHS(b^*)$. A sum $\sum$ of the hand-over cost x_b at each candidate b, where $b^* < b \le b^\times$, is thus an upper bound of $f(\cdot)$.

$$f(\cdot) = O\left(\sum_{b=b^*+1}^{b^\times} [b \in B] x_b\right), \tag{14}$$

where $[\cdot]$ is an Iverson bracket [10], being a 1 if the condition inside is met, otherwise 0. For the benifit of easy calculation, let a hand-over x_b starts from $\hat{p}_n = p^*$, $(\hat{p}_{LHS(b)} < p^*)$, and ends at $\hat{p}_{LHS(b)} = p^*$, $(\hat{p}_b > p^*)$. Then, by the fact that

$$\tilde{p}_{B_i} = \frac{\sum_{j=i-1}^{i+1} \tau_j^+}{\sum_{j=i-1}^{i+1} \tau_j}, \tag{15}$$

where at first $\tilde{p}_{LHS(b)}$ is approximate to $3w\mathcal{P}_{LHS(b)}$. And the end is $\tilde{p}_{LHS(b)}$ being p^* by sampling on b. A weighted calculation could discover how many should x_b be,

$$\begin{aligned} p^* &= \frac{3w\mathcal{P}_{LHS(b)} + x_b p_b}{3w + x_b} \\ x_b &= 3w\frac{p^* - \mathcal{P}_{LHS(b)}}{p_b - p^*}. \end{aligned} \tag{16}$$

The sum of all x is

$$\sum x_b \approx \sum_{k=b^*+1}^{b^\times} 3w\frac{p^* - \mathcal{P}_{LHS(k)}}{p_k - p^*}[k \in B]. \tag{17}$$

Let $r = \max\left((p^* - \mathcal{P}_{LHS(k)})/(p_k - p^*)\right)$, Eq. 17 is bounded by

$$\sum_{k=b^*+1}^{b^\times} 3wr[k \in B] = O\left(3wr\epsilon^{-1}\right) = O\left(r\epsilon^{-1}\right). \tag{18}$$

Notably, this is still a loose bound to some extent. Since for different n, x_b in Eq. 16 might differ a lot, the largest x_b is picked to act as the term r of the bound.

The exploitation part $g(\cdot)$ is the time complexity of termination after b^* is found. Supposed that b^* and one of its neighbor are the correct bars b_1 and b_2, i.e. $p_{b_1} \le p^* \le p_{b_2}$. The termination condition, by the first line of Eq. 12, is that p^* lies in $[p_{b_1}, p_{b_2}]$ with a sufficiently high probability. And a number of trials should be examined at $n = b_1$ and b_2, each numbers x_1, x_2. The cost of the exploitation phase $g(\cdot) = x_1 + x_2 = O(\max(x_1, x_2))$. Without loss of generality, let

$$|p^* - p_{b_1}| < |p^* - p_{b_2}|, \tag{19}$$

so that $x_1 > x_2$. The termination condition asks for $\mathbb{P}(p^* < p_{b_1}) + \mathbb{P}(p^* > p_{b_2}) < \delta$. According to Eq. 19, a stronger statement is $\mathbb{P}(p^* < p_{b_1}) < \frac{\delta}{2}$ if $x_2 \approx x_1$. By a proof similar to that of Section 4.2, the time complexity for $g(\cdot)$ is

$$O(-\Delta^2(\ln\delta - \ln 2)) = O(-\Delta^2 \ln\delta), \tag{20}$$

where $\Delta = |p^* - p_{b^*}|$. The sum of two phases together,

$$f(\cdot) + g(\cdot) = O\left(r\epsilon^{-1} + (-\Delta^2 \ln\delta)\right). \tag{21}$$

Comparing the complexity of GARL (Eq. 21) to that of PB (Eq. 6), if the exploration phase $f(\cdot)$ consumes rather little comparing to the exploitation phase $g(\cdot)$, which is experimentally fairly common, GARL will win by at least a factor of δ'/δ owing to the heuristic characteristic. PB, on the other hand, follows a deterministic approach so that a more severe residual is needed as a trade-off.

6 EXPERIMENTS

The experiments part is organized as follows: in Section 6.1, we compared the efficiency of precise bisection and our novel approach GARL for the simple GA (sGA) [5] on the OneMax problem; then in Section 6.2, in order to show its applicability, we applied GARL on several different GAs with a few optimization tasks; we also suggested the proper prior curve by the numerical method in Section 6.3.

6.1 Efficiency

The experiment in this subsection is twofold. First, to understand the influence of different problem size to the efficiency is the main focus. The experiment was set up as 25-, 50-, 100-, 200-bit OneMax problems (Eq. 22), while p^* was fixed to 0.93, N 150, ϵ 0.025, and δ 0.05. Second, we examined the situation for different N. The maximum population size N was set to $50 \times 2^i, i = 0, 1, \ldots, 9$ respectively. And the problem size was 50-bit while other parameter remained the same.

In both parts, we used a sGA using uniform crossover with exchange probability 0.5 and tournament selection with size 2 without replacement. As for the other hyperparameters, the was crossover rate 1, the mutation not used. Also, the max generation was set as 100.

$$f^{\text{OneMax}}(\boldsymbol{x}) = \sum_i \boldsymbol{x}_i \tag{22}$$

Figure 2 showed the efficiency comparison between PB and (ϵ-)GARL given different problem length. The data were analyzed under 30 runs for both algorithms. GARL outperformed PB in both average cost and the degree of variation. And also, the growth of GARL was experimentally slower than that of PB by an exponential factor. Besides, the error rate for both algorithms loosely agreed with the hyperparameter $\delta = 0.05$, which established the applicability.

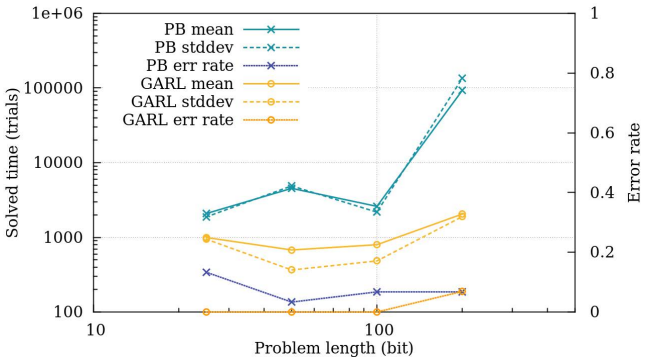

Figure 2: Problem size to efficiency and error rate plot.

Figure 3, in another facet, showed the N-to-efficiency plot between PB and (ϵ-)GARL. Since the larger the N, the bigger range $n^* \pm \epsilon N$ is, the terminate condition of a problem with larger N is therefore easier to be fulfilled. To a certain extent, it is so easy that just $O(1)$ toss at every candidate could reject the wrong candidate. So that the total tosses here for PB is in $O(-\lg \epsilon)$, and for GARL $O(\epsilon^{-1})$. In Figure 3, PB and GARL reached the point at $N = 3200$ and 1600 each.

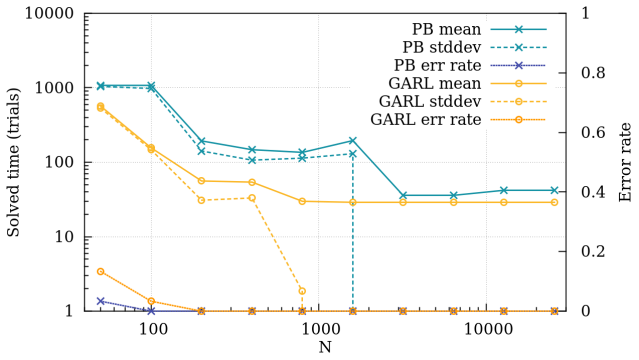

Figure 3: N to efficiency and error rate plot.

6.2 Applicability

In this section, we show the applicability of the work. Two simple experiments were chosen for the demonstration: the first is the MK trap where $k = 5$. MK trap is composed of m different blocks with k bits each [4]. For each block, without loss of generality, an all-1 sequence is the optimal, but other than that, the more 0, the higher the fitness. It is therefore called a trap function. The fitness function is shown in Eq. 23. Another point worth mentioning is we picked the r in Eq. 24 as 0.8.

$$f^{\text{trap}}_{m,k}(\boldsymbol{x}) = \sum_{i=1}^{m} f^{\text{trap}}_k \left(\sum_{j=ik-k+1}^{ik} \boldsymbol{x}_j \right), \tag{23}$$

where

$$f^{\text{trap}}_k(u) = \begin{cases} 1, & \text{if } u = k \\ r\left(1 - \frac{u}{k-1}\right), & \text{otherwise} \end{cases}. \tag{24}$$

The other experiment is the OneMax problem. OneMax is simply a bitwise summation of the chromosome $\boldsymbol{x}$, shown previously in Eq. 22.

Both experiments followed the same methodology: for each problem length, first the n^* was found using GARL where the hyperparameters was $p^* = 0.9, N = 200, \epsilon = 0.025, \delta = 0.05$ (See Table 1a and 1b). Secondly, the problem length and the population size were set as the parameters for each of the GAs: linkage tree GA (LTGA) [1], DSMGA-II [9], and simple GA respectively. For every set of parameters, 100 times of runs were performed and the mean NFE (of all successful runs), the standard deviation of the NFEs, and the error rate were recorded (See Figures 4 and 5).

Firstly, the MK-trap was examined. Only LTGA and DSMGA-II were tested in this part since the population size for sGA to solve the problem is way too large. It was a typical scenario to make use of the metric: two different GAs are judged by which GA would perform better on this task. Then a proper p^* was picked as 0.9, meaning that averagely 9 out of 10 trials would succeed. The result (Figure 4) suggested that DSMGA-II outperformed LTGA on this task with a lower average NFE. The experimental error rates also roughly conformed with $1 - p^*$ as well.

Length	25	50	100	200
LTGA	55	65	95	115
DSMGA-II	65	75	85	95

(a) n^* for MK trap, $k = 5$

Length	25	50	100	200
sGA	25	35	55	75
LTGA	5	15	15	15
DSMGA-II	5	5	5	5

(b) n^* for OneMax

Table 1: n^* for two tasks, $p^* = 0.9$, $N = 200$, $\epsilon = 0.025$ for both.

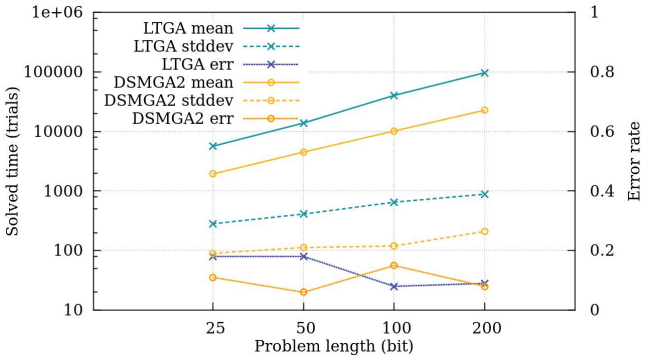

Figure 4: The NFE comparison on MK trap, where $k = 5$, population setting refers to table 1a.

The OneMax problem was examined afterward. This part (Figure 5) showed a not-so-desirable but discussible result. SGA was tuned exactly same as that of Section 6.1. Due to the largeness of N, the resolution of finding the correct n^* was actually not fine enough. So that for LTGA, population size being 5 was too small for $l = 25$, while 15 was too big for $l = 50, 100, 200$, resulted in a 64% error rate on the former and an average 4% error rate on the latter. The user therefore should consider limiting N or allowing a finer ϵ in this occasion. The mis-parameterization also influenced the average NFE of LTGA. Also, since DSMGA-II always performs a local hill-climbing after its initialization, it solves OneMax essentially with the least NFE possible, which is l (for hill-climbing by flipping each bit) plus a small number c. The error rate is always zero as well. So, it is barely comparable to others in this experiment.

6.3 Curve of the Prior Belief

Since if the prior belief of the population-size-to-convergence-rate curve $\mathcal{P}$ is closer to the real curve, $b^\times$ will be closer to b^*, and the hand-over process will also be faster, it is desirable to find an approximation to the real curve instead of linear prior. Through an empirical investigation, it is shown to be possible to use a logistic function (Eq. 25) to approximate the underlying n-to-p curve.

$$f(x) \stackrel{\text{def}}{=} \frac{1}{1 + \exp\left(-\frac{1}{a}(x - x_0)\right)}, \tag{25}$$

where a^{-1} is the steepness of the curve, and x_0 is the x-value of the midpoint. By regression, $\lg a, \lg x_0$, and $\lg l$ are propotionated on the same problem with the same GA. Figure 6 shows an example of the convergence rate of sGA and LTGA on OneMax problems. Therefore, if a and x_0 is known for the same problem and GA with

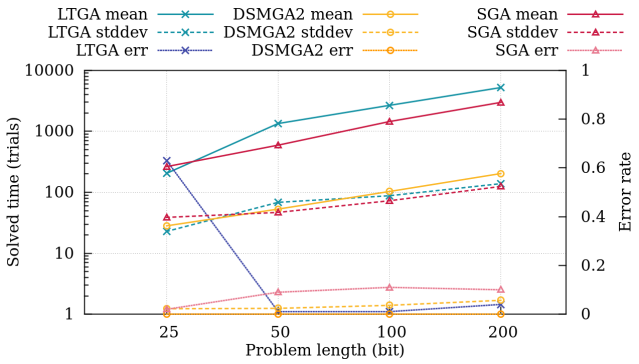

Figure 5: The NFE comparison on OneMax, population setting refers to table 1b.

the only difference being the length, then by calculation, the prior could be installed with a high weight w. Otherwise a logistic prior suggested by the regression of some known points or simply the linear prior will do the work. However, in this case, the weight w should be lower.

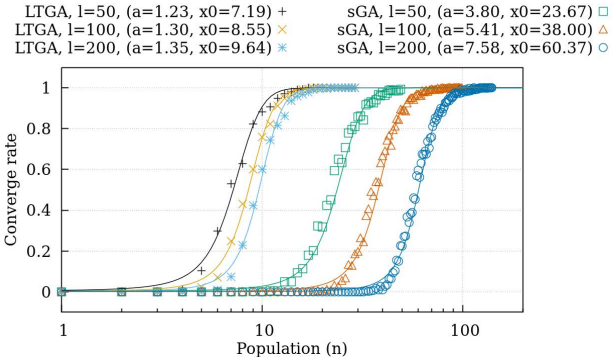

Figure 6: Population-to-convergence-rate plot of two GAs on OneMax problem and their regression.

7 CONCLUSION

Our work made clear the current measurement, NFE, of the efficiency of GA, formalized the metric, as well as proposed a cost-effective algorithm GARL for finding the most suitable population size being the basis for the fair comparison of various GAs. To the extent of our knowledge, this is the first ever work to formally address this problem.

Provided with the condition that the GA needs to converge to the global optimum with a certain predefined (implicit or explicit) probability, two existing ways to compare the efficiency of GA are as follows: (1) compare GAs by the minimum NFE to achieve the condition, or (2) firstly find the minimum population sizes for each GA to achieve the condition, and then compare GAs with the NFE consumed on those.

The techniques developed in this paper followed the second way. Given the fact that a larger population size yields a higher probability of successful convergence, we argued that the population-size

finding process can be more efficient in such methodology. In this research, we further formulated the idea of finding the population size for practical use. Specifically, we considered the confidence interval of the successful convergence rate by introducing some tolerance to the target population; the precision of the output and the maximum population size were taken into consideration as well.

Three population sizing algorithms were considered in this paper: the naïve bisection, the precise bisection (PB), and GARL. We first proved that the naïve bisection is biased and thus harms the fairness of the comparison. Then, a modified version, PB, was introduced, yielding unbiased results while consuming more function evaluations in general. Next, our new algorithm, GARL, was proposed, which follows a smoothed greedy practice. Notably, an installation of a prior belief is allowed in GARL, letting the procedure be even faster with the information of the population-size-to-convergence-rate curve provided. We also showed the time complexity of PB and GARL for an expected case, giving theoretical upper bounds to this problem. Both theoretical analysis and empirical results indicated that GARL outperforms the other two algorithms and is computationally efficient for researchers to compare their GAs.

Among all potential future research directions, two routes catch our interest. Firstly, we would like to investigate the smoothing function of GARL. Currently, the algorithm takes the samples from LHS and RHS neighbors into consideration to infer the expected convergence rate given a population size. But two aspects of this part have not been investigated yet: whether more neighbors should be taken into consideration instead of one on each side, which could also be described as a larger window size, and whether the samples from neighbors should have a less influence on the inferred expected value, realized by a weighted sum. Secondly, we would also like to derive a tighter complexity bound for GARL because several parts in the current derivation were overestimated.

REFERENCES

[1] Bosman, P. A., and Thierens, D. More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13* (2013), ACM Press.

[2] Cantor, G. Ueber eine Eigenschaft des Inbegriffs aller reellen algebraischen Zahlen. *Journal für die reine und angewandte Mathematik 1874*, 77 (1874), 258–262.

[3] Chang, C.-H., and Yu, T.-L. Investigation on parameterless schemes for DSMGA-II. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion - GECCO '16 Companion* (2016), ACM Press.

[4] Deb, K., and Goldberg, D. E. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence 10*, 4 (dec 1994), 385–408.

[5] Goldberg, D., and Holland, J. Genetic Algorithms and Machine Learning. *Machine Learning 3*, 2 (1988), 95–99.

[6] Goldman, B. W., and Punch, W. F. Parameter-less population pyramid. In *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14* (2014), ACM Press.

[7] Harik, G. R., and Lobo, F. G. A parameter-less genetic algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1* (1999), Morgan Kaufmann Publishers Inc., pp. 258–265.

[8] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*, 301 (mar 1963), 13–30.

[9] Hsu, S.-H., and Yu, T.-L. Optimization by pairwise linkage detection, incremental linkage set, and restricted / back mixing. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15* (2015), ACM Press.

[10] Iverson, K. E. A programming language. In *Proceedings of the May 1-3, 1962, spring joint computer conference on - AIEE-IRE '62 (Spring)* (1962), ACM Press.

[11] Mühlenbein, H., and Paass, G. From recombination of genes to the estimation of distributions i. binary parameters. In *International conference on parallel problem solving from nature* (1996), Springer, pp. 178–187.

[12] Pelikan, M. *Hierarchical Bayesian Optimization Algorithm*. Springer Berlin Heidelberg, 2005.

[13] Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1* (San Francisco, CA, USA, 1999), GECCO'99, Morgan Kaufmann Publishers Inc., pp. 525–532.

[14] Pelikan, M., Goldberg, D. E., and Lobo, F. G. A survey of optimization by building and using probabilistic models. *Computational optimization and applications 21*, 1 (2002), 5–20.

[15] Pelikan, M., and Lin, T.-K. Parameter-less hierarchical boa. In *Genetic and Evolutionary Computation – GECCO 2004* (Berlin, Heidelberg, 2004), K. Deb, Ed., Springer Berlin Heidelberg, pp. 24–35.

[16] Sastry, K. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2001. Also IlliGAL Report No. 2002004.

[17] Thierens, D., and Bosman, P. A. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11* (2011), ACM Press.

[18] Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[19] Yu, T.-L., Goldberg, D. E., Sastry, K., Lima, C. F., and Pelikan, M. Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary Computation 17*, 4 (2009), 595–626. PMID: 19916779.